View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Repositorio Documental de la Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

MENCIÓN EN TELEMÁTICA

Aplicación de algoritmos de Machine Learning para la predicción del beneficio por cliente a partir de métricas de Google Analytics

Autor:

D. Pablo Eliseo Gonzalo Fuentes

Tutor:

Dr. Ignacio de Miguel Jiménez

Valladolid, Septiembre de 2019

TÍTULO:	**Aplicación de algoritmos de Machine Learning para la predicción del beneficio por cliente a partir de métricas de Google Analytics.**
AUTOR:	**D. Pablo Eliseo Gonzalo Fuentes**
TUTOR:	**Dr. Ignacio de Miguel Jiménez**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática.**

TRIBUNAL

PRESIDENTE:	**Evaristo J. Abril Domingo**
SECRETARIO:	**Juan Carlos Aguado Manzano**
VOCAL:	**Ignacio de Miguel Jiménez**
SUPLENTE 1:	**Ramón J. Durán Barroso**
SUPLENTE 2:	**Noemí Merayo Álvarez**

FECHA:

CALIFICACIÓN:

Agradecimientos:

En primer lugar, al Dr. Ignacio de Miguel Jiménez por su apoyo, consejos y seguimiento durante la realización de este Trabajo de Fin de Grado.

A mis amigos y compañeros que me han apoyado en todo momento, escuchándome y dándome su punto de vista. En especial a Javier Olmedo, compañero con quien he ido comentando resultados durante todo el proceso.

Por último, a mis padres y hermana, quienes han mostrado interés y ayuda, y sobre todo porque sin ellos no habría llegado hasta aquí.

Resumen

Google Analytics es una potente herramienta de analítica web, que permite conocer datos muy detallados sobre las características de las visitas a un sitio web. Esta información puede ser utilizada por las empresas para conocer qué perfil de usuario proporciona mayores beneficios o para predecir el beneficio esperado de un cliente a partir de la historia previa (tanto de ese como de otros clientes).

El objetivo de este Trabajo de Fin de Grado consiste en utilizar diferentes bibliotecas de machine learning en Python para predecir los ingresos que se obtendrán por cliente a partir de un conjunto de datos de compras anteriores. En primer lugar, hemos obtenido uno o varios conjuntos de datos de Google Analytics y hemos realizado un análisis de dichos datos para su comprensión. A partir de ahí, se ha elaborado una serie de modelos predictivos de los ingresos por cliente, y se han evaluado sus prestaciones para la elección del mejor de ellos.

Palabras clave:

Python, Análisis masivo de datos, Aprendizaje automático, Sitio web, Analítica web.

Abstract

Google Analytics is a powerful tool used for web analytics, which allows you to know highly detailed information about the characteristics of the visits to a website. This information can be used by companies to know which user profile provides the greatest benefits or to predict the expected benefit from a client based on his/her previous history.

The objective of this Bachelor Thesis is to use different machine learning libraries of Python to predict the revenue that will be obtained per client from a set of data from previous purchases. First, one or several data sets of Google Analytics has been obtained and an analysis of that data has been performed for its understanding. From there, some predictive models of revenue per client has been developed, and its benefits has been evaluated for the choice of the best model.

Palabras clave:

Python, Big Data, Machine Learning, Website, Web analytics.

Índice general

Capítulo 1. Introducción 1

1.1 Motivación y Objetivos 1

1.1.1 Aparición y evolución de las herramientas de analítica web 1

1.1.2 Funcionamiento de las herramientas de analítica web 4

1.1.2.1 Métricas clave 5

1.1.2.2 Google Analytics 7

1.1.3 Objetivo 8

1.2 Fases y métodos 8

1.3 Medios disponibles 10

1.3.1 Software 10

1.3.1.1 Python 10

1.3.1.2 Numpy 11

1.3.1.3 Pandas 11

1.3.1.4 Plotly 11

1.3.1.5 Scikit-Learn 11

1.3.1.6 Google Cloud Platform 12

1.4 Estructura de la Memoria 12

Capítulo 2. Aprendizaje Supervisado 14

2.1 Introducción 14

2.2 Funcionamiento del Aprendizaje Supervisado 15

2.3 Teoría 16

2.3.1 Ejemplo con regresión lineal 17

2.3.1.1 Regularización 20

2.3.1.2 Validación 22

2.3.1.3 Normalización 23

2.3.2 Otros modelos interesantes 24
2.3.2.1 Árbol de decisión 24
2.3.2.2 Máquinas de vectores de soporte 26
2.3.2.3 Red Neuronal Prealimentada (*Feedforward neural network*) 27
Capítulo 3. Análisis de los Datos 29
3.1 Introducción a los datos 29
3.2 Exposición de los datos disponibles 29
3.2.1 Tablas 29
3.2.2 Filas 30
3.2.3 Columnas 30
3.3 Análisis de los datos 32
3.3.1 Análisis de columnas 32
3.3.1.1 Columnas independientes 33
3.3.1.2 Totals 33
3.3.1.3 TrafficSource 34
3.3.1.4 Device 35
3.3.1.5 GeoNetwork 36
3.3.2 Análisis por filas 37
3.3.2.1 Totals 37
3.3.2.2 TrafficSource 38
3.3.2.3 Device 38
3.3.2.4 GeoNetwork 38
3.3.3 Creación de características 40
3.3.4 Set de datos final 41
Capítulo 4. Desarrollo de Modelos 42
4.1 Preprocesado 42
4.2 Elección de Modelo 42

Capítulo 5. Conclusiones y Líneas Futuras 46

5.1 Conclusiones 46

5.2 Líneas futuras 47

REFERENCIAS 48

Preparación previa 52

5.3 Cursos 52

5.3.1 Curso de introducción a Google Analytics 52

5.3.2 Curso avanzado de Google Analytics 52

5.3.3 Curso de programación en Python 52

5.3.4 Curso introducción al Machine Learning 53

5.3.5 Curso intermedio de Machine Learning 53

5.3.6 Curso de Pandas 53

Índice de figuras

Ilustración 1.1 Primera página web de la historia. Disponible en http://info.cern.ch/hypertext/WWW/TheProject.html 1

Ilustración 1.2 Contadores de visitas 2

Ilustración 1.3 Interfaz gráfica de Google Analytics 7

Ilustración 1.4 Esquema de metodología CRISP-DM 10

Ilustración 2.1 Esquema de diferentes tipos de Aprendizaje Automático 14

Ilustración 2.2 Representación de modelo genérico (Bonaccorso, G. et al 2018) 16

Ilustración 2.3 Diagrama de hipótesis final (Abu-Mostafa, Y. S. et al 2012)..... 17

Ilustración 2.4 Punto óptimo del modelo 19

Ilustración 2.5 Regresión lineal con datos no linealmente separables 19

Ilustración 2.6 Comparativa de modelo con y sin regularización 21

Ilustración 2.7 Separación de datos en sets de entrenamiento y validación..... 22

Ilustración 2.8 Diferencia entre set de datos original y set de datos centrado en cero..... 23

Ilustración 2.9 Ejemplo sencillo de árbol de decisión (Machine Learning Mastery, 2017). 24

Ilustración 2.10 Red Neuronal Prealimentada 28

Ilustración 3.1 Valores vacíos en columnas independientes 33

Ilustración 3.2 Valores vacíos en registro totals 34

Ilustración 3.3 Valores vacíos en registro trafficSource 35

Ilustración 3.4 Valores vacíos en registro device 36

Ilustración 3.5 Valores vacíos en registro geoNetwork 36

Ilustración 3.6 Matriz de correlaciones del registro geoNetwork 39

Ilustración 3.7 Filas con datos conflictivos en el registro geoNetwork 40

Ilustración 4.1 Número de valores 0.0 que tiene la etiqueta totals.totalTransactionRevenue 43

Ilustración 4.2 Comparativa de modelos en gráfico de cajas con métrica de error Root Mean Squared Error 44

Ilustración 4.3 Comparativa de Redes Neuronales en gráfico de cajas con métrica de error Root Mean Squared Error 44

Índice de tablas

Tabla 1.1 Métricas clave 6

Tabla 3.1 Campos de las tablas de exportación de Google Analytics (Google Support, 2018). ... 32

Capítulo 1. Introducción

Este primer capítulo trata sobre el marco en el que se ubica este Trabajo de Fin de Grado, detallando su motivación, el objetivo final y, por último, las herramientas utilizadas.

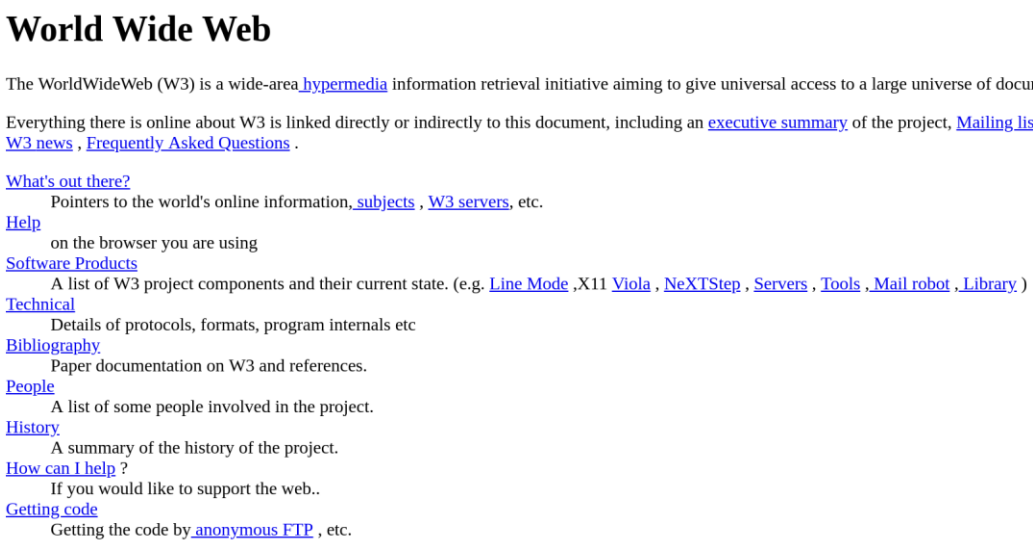
1.1 Motivación y Objetivos

Para ubicar el marco de este documento, debemos remontarnos a la historia de Internet y las analíticas web.

1.1.1 Aparición y evolución de las herramientas de analítica web

En 1990, en algunas de las salas del CERN de Suiza, el equipo de Sir Tim Berners-Lee usaba una red de ordenadores para enlazar los documentos entre sí y así poder trabajar de forma colaborativa. Apoyándose en el protocolo TCP/IP, el equipo de Berners-Lee solo tenía que hacer click en los denominados hipertextos para acceder de forma automática a los documentos deseados.

Utilizando un protocolo de transferencia de hipertexto (HTTP), un lenguaje de marcado (HTML), un servidor que soporta esta tecnología, el primer FTP y el primer navegador web, bastante más sencillo que los que conocemos ahora y llamado WorldWideWeb, se daba a conocer en esas fechas con la primera página web de la historia (Ilustración 1.1).



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Ilustración 1.1 Primera página web de la historia. Disponible en http://info.cern.ch/hypertext/WWW/TheProject.html

En ese mismo instante acababa de nacer la web y con ello la mayor revolución global que nuestra generación ha conocido.

Este sistema de colaboración permitía enlazar cualquier documento con otro que estuviera en la misma red, lo cual atrajo la atención de muchos interesados en compartir y acceder a estas fuentes de información.

Durante los primeros años fueron, en su mayoría, departamentos científicos de universidades y laboratorios los que hicieron uso de esta tecnología, sin embargo, poco a poco se fue extendiendo a otros ámbitos, lo cual generó nuevas necesidades en los documentos, como imágenes o ciertos comportamientos dinámicos.

En 1993 Webtrends publica una web considerada la primera herramienta comercial de análisis de datos en la Web. Esta herramienta surge de la necesidad de los usuarios de conocer la máxima información posible acerca del uso de estos documentos subidos a la Web (Amplitude, 2015).

En aquella época los sitios web no eran más que imágenes y texto, WebTrends se encargaba simplemente de ordenar la información extraída de los logs de los servidores web. En estos logs se almacenaban métricas tales como consumo de contenido, visitas y duración de las sesiones, siendo cada línea de texto una interacción con el documento. WebTrends era capaz de representar esta información de manera comprensible para el ser humano, aunque fueran herramientas pensadas para ser únicamente usadas por informáticos.

En años posteriores aparecieron diferentes herramientas, algunas muy interesantes, tales como NetGenesis, ahora integrada en SPSS (Dealipedia, 2007). Sin embargo, la herramienta realmente importante, que supuso un salto realmente cualitativo fue Analog, creada por Stephen Turner en 1995 (analog.cx). Analog abrió la analítica web a los profesionales del marketing mostrando informes completos y comprensibles que incluían **gráficas** donde percibir visualmente tendencias y desviaciones (Gorostiza, I. et al 2016).

Después de esto, en 1996, la analítica web se popularizó de forma masiva, motivado por la aparición de Web-Counter, el primer contador de visitas (Ilustración 1.2). Se trataba de un pequeño programa que permitía a los sitios web mostrar el número de visitas (en realidad páginas vistas) que habían recibido (Gorostiza, I et al 2016).

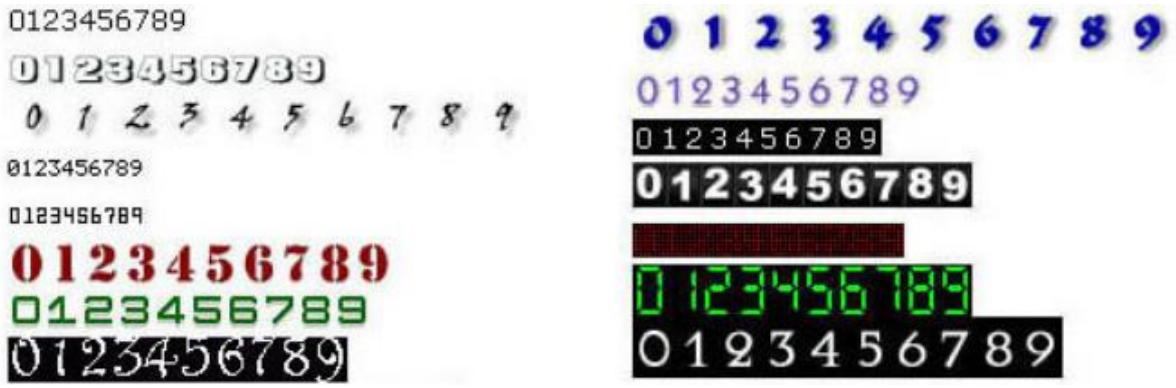


Ilustración 1.2 Contadores de visitas

En ese mismo año nacieron empresas como Accrue, Omniture o WebSideStory.

Internet seguía creciendo, cada vez era más popular y esto conllevó a una mayor complejidad de las páginas web. El análisis de los logs permitía hacer exclusivamente un seguimiento de las peticiones al servidor, pero dentro de las páginas se empezaron a introducir interacciones que no podían ser registradas: contenidos dinámicos, multimedia, efectos visuales… Las herramientas de analítica web tenían que adecuarse a esta nueva realidad.

En 1997 surgió un nuevo método de recolección de datos basado en un código de seguimiento, escrito en JavaScript, que permitía recoger todas las acciones del usuario en la página sin que éste tuviera que realizar una interacción con el servidor.

El sistema de seguimiento por JavaScript se hizo muy popular, de hecho, sigue siendo utilizado hoy en día por muchas de las herramientas más populares (entre ellas Google Analytics).

Internet ya formaba parte de nuestras vidas y por el año 2000 un gran número de empresas empiezan a florecer en la red. Empresas de capital riesgo invirtieron importantes cantidades de dinero en startups.

La industria no tenía aún la madurez suficiente, lo cual provoca la llamada "burbuja de las .com", lo cual provocó un parón en el avance de la industria de la analítica web debido al recelo que sentían muchas compañías a apostar por estas nuevas tecnologías.

Los años siguientes, empresas de renombre como Google, Amazon o eBay dispararon el uso de la web en todo el mundo, comenzando así a integrarse en nuestra sociedad. Además, comenzaron a popularizarse nuevas formas de comunicación por medio de la Web como los blogs, redes sociales, etc.

La web ya era un canal fundamental de comunicación, venta y una increíble fuente de información. Conocer a fondo el funcionamiento de los sitios web y el comportamiento de los usuarios en ellos pasó de ser una mera curiosidad para convertirse en una necesidad y un negocio en sí mismo.

En 2004 nace la asociación de analítica web (WAA) en la cual se reúnen profesionales del sector y foro donde pueden debatir sus necesidades e inquietudes.

En marzo de 2005 Google compra la empresa Urchin. Esta empresa era la mejor posicionada y la que mejor estaba gestionando el apartado gráfico y visual de sus informes. Unos meses después de comprar Urchin, tras un lavado de cara de la interfaz de usuario, Google lanza en noviembre de 2005 Google Analytics.

La aparición de herramientas de *in-page analytics* como ClickTale en 2006 permitió un análisis más profundo del comportamiento de los usuarios, mostrando así el análisis cuantitativo de una forma más clara (Amplitude, 2015).

Herramientas que proporcionaban mapas de calor, interacciones de scroll, incluso algunas que grababan las sesiones de los usuarios empezaron a popularizarse durante los procesos de análisis para la optimización de los sitios web.

En paralelo, Google Analytics se consolidaba como la herramienta líder del mercado. Una interfaz intuitiva, combinada con una gran cantidad de informes de gran calidad, una sencilla implementación y el hecho de ser totalmente gratuita provocó que la mayoría de los usuarios se decantaran por ella, dejando de lado a sus competidoras, que, aunque potentes, eran más difíciles de configurar y, sobre todo, más caras.

La cantidad y calidad de las herramientas fue creciendo a pasos agigantados y, en 2010, surgió un importante cambio. Google Analytics cambió su código de seguimiento, el que gestiona la

herramienta, por un código asíncrono. Este cambio de paradigma facilitó una mejor gestión del etiquetado, obteniendo una mayor fiabilidad en los datos registrados. El código cargaba más rápido, con menos errores y además se obtenían resultados más fieles a la realidad.

Un año después, la influencia que las redes sociales tenían obligó a que todas las herramientas buscaran fórmulas para mostrar la actividad que se producía ahí. Además, los usuarios se conectaban a Internet con múltiples dispositivos, los móviles implicaban una continua conectividad de los usuarios, así pues, las herramientas basadas en JavaScript y las denominadas First-Party cookies comenzaron a buscar soluciones al problema del multi-dispositivo, que imposibilitaba la identificación única de los usuarios reales.

Para intentar dar respuesta a estas nuevas necesidades Google dio un giro a su herramienta, lanzando en 2013 Google Universal Analytics.

Google Universal Analytics implica un nuevo concepto de medición basado en un código de seguimiento que trae nuevas funcionalidades que permiten, entre otras cosas, asignar a los usuarios un identificador único y mantenerlo a través de dispositivos y navegadores.

Esta herramienta realiza un seguimiento intensivo del usuario, mostrando todas las interacciones que este realiza con el sitio web, además de diferentes métricas que veremos más adelante.

1.1.2 Funcionamiento de las herramientas de analítica web

Las analíticas web miden métricas como las vistas de página, las visitas, los visitantes únicos, etc. El resultado del análisis muestra tanto la cantidad como la calidad de los visitantes de la web.

La finalidad de las analíticas web es proveer información de estos visitantes para entender cómo está siendo usado un determinado sitio web y aplicar esta información para optimizar su uso. Es más que una colección de datos, es un intento de entender cómo la gente usa un sitio web y por qué.

Con una buena herramienta de analítica web, se pueden obtener datos como:

- Cuántos visitantes atrae el sitio web.
- Cuál es la procedencia de estos visitantes. Qué sitios direccionan la mayor parte de tráfico a dicha web, además de su situación geográfica.
- Por cuánto tiempo están los visitantes en el sitio web.
- Qué páginas visitan en primera instancia sus visitantes y cuáles visitan justo antes de irse.
- Si los visitantes provienen de una búsqueda orgánica (como Google, Yahoo…), qué palabras clave buscaron para acabar en la página web.
- Si los visitantes provienen de un anuncio, dónde está situado el anuncio y el porcentaje de visitantes que hicieron clic en el anuncio y visitaron páginas o completaron transacciones.
- Qué tipo de navegadores y dispositivos están utilizando los visitantes de la web.

Cuando se trata de rastrear visitantes de la web, hay dos tipos de analítica fundamentales (Miller, M. et al 2011):

- El análisis “Onsite”: Utiliza datos específicos del sitio. Utiliza una técnica llamada *etiquetado de página*. Esta técnica introduce un código JavaScript en el HTML de la página. Este código embebido recolecta información sobre la página y sus visitantes, la cual es pasada a la web del servicio de analíticas, que la coteja y la utiliza para crear varios reportes analíticos.
- El análisis “Offsite”: Utiliza información a lo largo de Internet para determinar los sitios más visitados de la web. Se utiliza para compilar el análisis de toda la industria, mientras que la analítica “onsite” se utiliza para informar sobre el rendimiento del sitio web individual. Los propietarios de sitios web y los webmasters están más interesados en el análisis “onsite”.

1.1.2.1 Métricas clave

Gracias a los datos obtenidos con el código JavaScript, los servicios de analíticas web crean una serie de métricas que pueden ayudar a conocer mejor a los visitantes de la misma. En la Tabla 1.1 se pueden ver los detalles de las métricas más importantes a medir (Miller, M. et al 2011):

Métrica	**Descripción**
% exit	El porcentaje de usuarios que salen de una página web dada como una parte de las páginas vistas.
Bounce rate	El porcentaje de visitas en las que el visitante entra y sale en la misma página, sin visitar ninguna otra página en el sitio.
Click	Una sola instancia de un visitante haciendo clic en un enlace de una página a otra en el mismo sitio.
Click path	La secuencia de clics que los visitantes del sitio web siguen en un sitio determinado.
Click-through rate (CTR)	El porcentaje de personas que ven un elemento y luego hacen clic en él; se calcula dividiendo el número de clics por el número de impresiones.
Depth of visit (pageviews per session)	El número promedio de páginas vistas que un visitante inicia antes de finalizar su sesión; se calcula dividiendo el número total de visitas de página por el número total de sesiones.
First Visit	La primera visita de un visitante que no ha visitado el sitio anteriormente
Hit	Una petición de un fichero a un servidor web. Un hit no es lo mismo que un “pageview”; una página puede tener múltiples elementos (imágenes, cajas de texto…) que necesitan ser descargados del servidor de manera individual.

Impression	Una sola visualización de un anuncio en una página web.
Loyalty	Una medida de la frecuencia con la que los visitantes vienen a un sitio web, calculado dividiendo el número total de sesiones o visitas por el número total de visitantes únicos.
New visitor	Un visitante que no ha hecho visitas previas al sitio web.
Pageview	Una visualización de una página web completa. Un visitante que mira una sola página en su sitio genera una vista de página. (Las vistas de página normalmente no incluyen páginas de error, o las páginas que ven los rastreadores web o robots).
Pageview duration	La cantidad promedio de tiempo que los visitantes pasan en cada página de un sitio web.
Repeat visitor	Un visitante que haya realizado al menos una visita previa a un sitio web.
Session	Una serie de visitas de página del mismo visitante con no más de 30 minutos entre visitas de página y sin visitas a otros sitios entre visitas de página. A diferencia de una visita, una sesión termina cuando un visitante abre una página en otro sitio.
Single page visit	Una visita de un visitante donde solo se ve una sola página.
Time on site or length of visit	La cantidad promedio de tiempo que los visitantes pasan en un sitio web cada vez que visitan.
Unique visitor	Un visitante que visita su sitio una o más veces dentro de un período de tiempo determinado, generalmente un período de 24 horas; un visitante puede realizar múltiples visitas durante ese período de tiempo, pero esto cuenta como un solo visitante único. Por ejemplo, un usuario que visita su sitio dos veces en un día se cuenta como un único visitante único.
Visit	Una serie de vistas de página del mismo visitante con no más de 30 minutos entre cada visita de página. A diferencia de una sesión, una visita continúa (durante 30 minutos) incluso después de que un visitante abandone su sitio.

Tabla 1.1 Métricas clave

1.1.2.2 Google Analytics

Google Analytics es una herramienta de analítica web muy completa en las métricas que rastrea. También es relativamente fácil de usar y completamente gratis.

Debido a su coste (gratis) Google Analytics es popular entre los sitios web grandes y pequeños. Google Analytics es lo suficientemente potente como para rastrear el tráfico en sitios web grandes, pero lo suficientemente fácil como para que lo implementen los sitios más pequeños. Rastrea todas las métricas clave detalladas en la Tabla 1.1 y más, y muestra los resultados en una serie de "Cuadros de mando" e informes personalizados, como el que se muestra en la Ilustración 1.3.

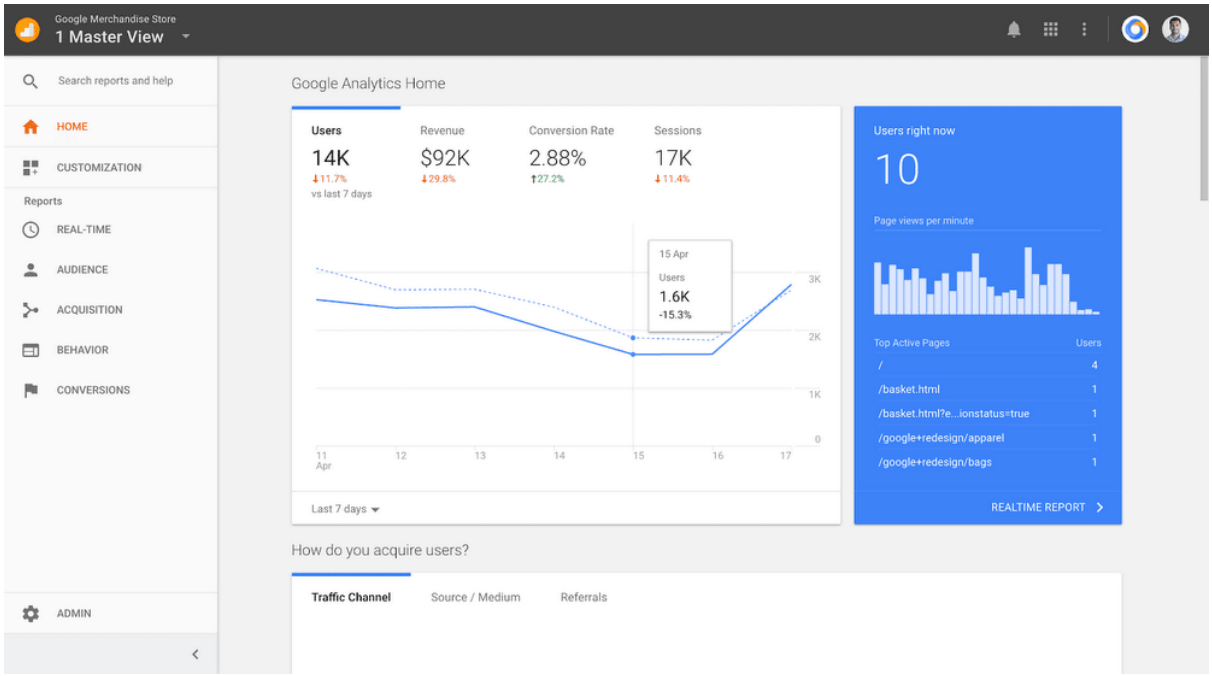


Ilustración 1.3 Interfaz gráfica de Google Analytics

Google Analytics utiliza analíticas “onsite” para rastrear el comportamiento de los visitantes en un sitio específico. Después de registrarlo con Google Analytics, Google genera una pieza única de código JavaScript para su sitio web. Luego, se debe copiar y pegar ese código en el HTML subyacente de cada sitio en la página que se desea rastrear. Una vez incorporado, este código rastrea el comportamiento de los visitantes y transmite esos datos a Google, donde se analiza y se muestra.

Todo este proceso de recolección de datos se desarrolla en 4 fases:

- **Fase 1: Recolección de datos:** La información enviada a Google Analytics a partir del código de seguimiento JavaScript embebido es recibida en forma de paquetes o “Hits” (interacciones). El código de seguimiento envía toda la información relacionada con la página/pantalla que el usuario está viendo. Recoge información del navegador y el dispositivo mediante el cual el usuario está accediendo. Además, incluye toda esta información sobre la fuente de tráfico de la cual procede el visitante. En lo que a confidencialidad se refiere, Google Analytics tiene las siguientes consideraciones:
    + Google Analytics no almacena ni reporta ningún tipo de información personal.
    + El personal de Google solo puede acceder a tus datos con tu permiso.

    + Se puede seleccionar la forma de compartir tu información con Google y otros proveedores.
- **Fase 2: Procesamiento de la información:** El procesamiento conlleva la transformación de la información en datos útiles y categorizados. En primer lugar, todos los paquetes o hits se organizan según sesiones y usuarios. Después, se junta la información recolectada por el snippet de seguimiento con la recogida por Google Ads, Adsense, Search Console y otras fuentes de datos externas a Google. Existen dos maneras de importar datos externos en Google Analytics:
    1. *Custom Data Import*: Permite importar nuevas dimensiones a los informes de Google Analytics subiendo un fichero o utilizando la API de Google Analytics.
    2. *Cost Data Import*: Se utiliza habitualmente para reportar a Google Analytics el dinero gastado en anuncios que no sean de Google Ads. Para ello hay que subir un fichero que incluya el medio y fuente de la campaña.
- **Fase 3: Configuración:** En esta fase se aplica cualquier personalización que hayamos planificado para la generación de informes. Se hace en casos como por ejemplo si se ha incluido un filtro para excluir tráfico interno de la red o para cambiar la apariencia de las urls. Finalmente, la información pasa por un proceso de agregación en el que se prepara para que pueda ser analizada. En este proceso los datos se organizan y almacenan en tablas de bases de datos para que se pueda acceder a ellos de manera rápida cuando sea necesario. Una vez procesada la información se almacenará en sus bases de datos y no podrá volver a ser modificada. Este es el motivo por el que nunca se podrán alterar los datos del histórico de Google Analytics.
- **Fase 4: Generación de Informes:** Al final se tiene acceso a todos estos datos en forma de informes. En nuestro caso de estudio, esta fase no es muy importante, dado que trabajaremos con los datos en crudo.

Cabe destacar que podremos exportar los datos de diferentes maneras. En este Trabajo de Fin de Grado se utilizan datos exportados a formato *.csv* y datos exportados a una tabla de Google BigQuery (serán los mismos datos, pero trabajaremos con ellos desde dos fuentes).

1.1.3 Objetivo

Como hemos visto, estas herramientas de analítica web ofrecen una inmensa cantidad de información, que en su gran medida es utilizada para la optimización de los sitios web de las empresas (o personas) que hacen uso de ellas. En este Trabajo de Fin de Grado hemos pensado en aplicar algoritmos de machine learning para obtener un resultado que (al menos en principio) dista bastante de esa finalidad; nuestra idea es utilizar estos datos para hacer una predicción del comportamiento de los clientes en un sitio web, siendo nuestro objetivo predecir los ingresos que se obtendrán de sus visitas, basándonos en la calidad de sus sesiones, o lo que es lo mismo, basándonos en su comportamiento en dicho sitio web.

1.2 Fases y métodos

Las fases en las que hemos dividido este Trabajo de Fin de Grado se exponen a continuación:

1. **Búsqueda de sets de datos:** Focalizada en encontrar una exportación de los datos de Google Analytics en un formato que nos resultara útil.
2. **Instalación del entorno de trabajo:** Como para cualquier proyecto de características similares, hemos tenido que adecuar nuestro entorno de trabajo con las herramientas software necesarias.
3. **Estudio del lenguaje Python, BigQuery y librerías necesarias:** Para realizar el análisis de los datos, así como el modelado y su evaluación, hemos tenido que familiarizarnos y aprender a utilizar las herramientas previamente mencionadas. Esto lo detallaremos en el ANEXO I.
4. **Conocimiento de la herramienta:** Para poder hacer un procesado correcto de los datos es imprescindible familiarizarse con el uso de la herramienta de la cual los hemos extraído. Para ello he realizado una serie de cursos de la Google Analytics Academy de los cuales he obtenido dos certificados.
5. **Comprensión de los datos:** Una vez conocemos la herramienta Google Analytics, debemos explorar la recopilación de los datos, su descripción y verificar su calidad.
6. **Preparación de los datos:** Antes de generar un modelo predictivo, debemos analizar el dataset, ver qué datos pueden ser útiles y cuáles no, limpiarlos en caso de ser necesario, y ponerlos en un formato adecuado.
7. **Modelado:** Empezando por un modelo sencillo, pasando por otros más complejos y ajustándolos de la forma necesaria para obtener mejores resultados.
8. **Evaluación:** Realizamos la evaluación de los resultados obtenidos con los modelos de la fase anterior.

En la diferenciación de estas fases (sobre todo de la 4 a la 8) nos hemos basado en la ampliamente utilizada metodología CRISP-DM[1] (*Cross-Industry Standard Process for Data Mining*).

[1] Modelo estándar abierto del proceso que describe los enfoques comunes que utilizan los expertos en minería de datos, más información en http://www.datascience-pm.com/crisp-dm-and-kdd/

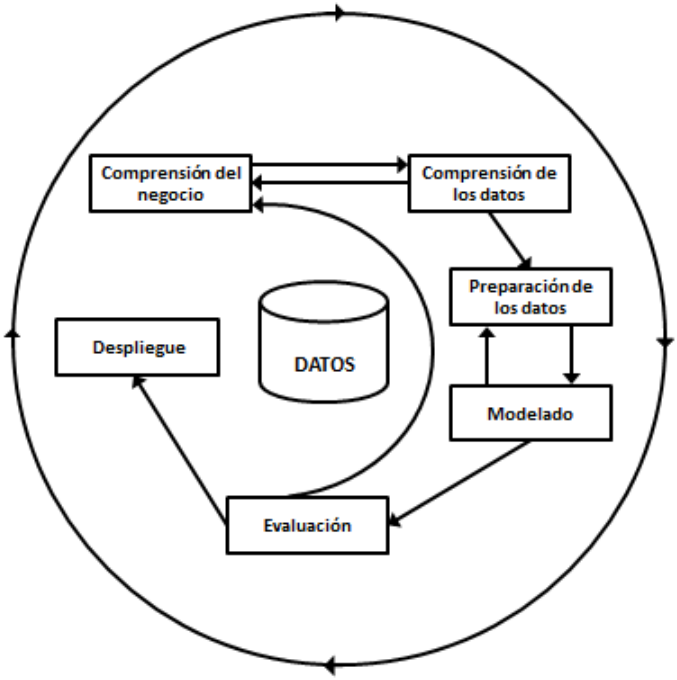


Ilustración 1.4 Esquema de metodología CRISP-DM

Cómo podemos ver en la Ilustración 1.4, muchas de las fases de la metodología CRISP-DM no son secuenciales, sino que, en función de sus resultados, implican volver a fases anteriores a realizar modificaciones o una nueva exploración para mejorar el resultado final.

También cabe darse cuenta de que, aun siguiendo esta metodología, hemos prescindido de la última fase (despliegue).

1.3 Medios disponibles

Este Trabajo de Fin de Grado se ha realizado, fundamentalmente, empleando un ordenador personal. Nos centraremos en el software necesario.

1.3.1 Software

1.3.1.1 Python

El lenguaje principalmente utilizado en este Trabajo de Fin de Grado ha sido Python.

Python es un lenguaje interpretado, multiparadigma (soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional), que usa tipado dinámico y cuya filosofía se basa en una sintaxis que favorezca un código legible.

Sus principales atractivos son que es relativamente fácil de aprender, comparado con sus competidores, y es portable, es decir, puede ser interpretado por casi todos los Sistemas Operativos, entre ellos UNIX, Mac OS, MS-DOS, 0S/2 y Windows (Python, 2019).

Fue creado por Guido van Rossum y el código es de libre acceso.

La característica más notable de este lenguaje es su indentado en las sentencias de fuente para facilitar la lectura. Python ofrece también tipos de datos dinámicos, clases de lectura e interfaces para llamadas al sistema y bibliotecas. Puede ser extendido usando C o C++.

En este Trabajo de Fin de Grado hemos utilizado la versión 3.7 de este lenguaje.

1.3.1.2 Numpy

NumPy es el paquete fundamental para la computación científica con Python. Contiene entre otras cosas:

- Un poderoso objeto de matriz N-dimensional
- Funciones sofisticadas (difusión)
- Herramientas para la integración de código C / C ++ y Fortran.
- Álgebra lineal útil, transformada de Fourier y capacidades de números aleatorios.

Además de sus obvios usos científicos, NumPy también puede usarse como un eficiente contenedor multidimensional de datos genéricos. Se pueden definir tipos de datos arbitrarios. Esto permite que NumPy se integre a la perfección con una amplia variedad de bases de datos (Numpy 2019).

1.3.1.3 Pandas

Pandas es una biblioteca de software escrita como extensión de NumPy para manipulación y análisis de datos para el lenguaje de programación Python. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales. Es un software libre distribuido bajo la licencia BSD. El nombre deriva del término "datos de panel", término de econometría que designa datos que combinan una dimensión temporal con otra dimensión transversal (Pandas 2019).

Es ampliamente utilizado en ciencia de los datos para manipular los datos en *dataframes*, instancias de su clase más característica.

1.3.1.4 Plotly

La biblioteca de gráficos Plotly crea gráficos en línea interactivos con calidad de publicación. Está disponible en diversos lenguajes, tales como Python, R, Matlab y Scala. En este Trabajo de Fin de Grado hemos utilizado esta librería debido a su calidad de gráficos y sobre todo a su capacidad de interacción con el usuario y su forma de exportar datos, facilitada al basarse en JavaScript.

1.3.1.5 Scikit-Learn

Scikit-learn es una biblioteca para aprendizaje automático de software libre para el lenguaje de programación Python. Incluye varios algoritmos de clasificación, regresión y análisis de grupos entre los cuales están máquinas de vectores de soporte, bosques aleatorios, Gradient boosting, K-means y DBSCAN. Está diseñada para interoperar con las bibliotecas numéricas y científicas NumPy y SciPy.

1.3.1.6 Google Cloud Platform

Google Cloud Platform es una suite que contiene diversos servicios que funcionan en la misma infraestructura que utiliza Google de manera interna. Dentro del conjunto de herramientas que Google ofrece nosotros hemos utilizado:

- **Compute Engine**: Plataforma que permite dotar de poder computacional a las máquinas en el cloud. Compute Engine se centra en tener una infraestructura como servicio, en la cual tenemos que configurar cada aspecto de esta infraestructura y hacer la gestión de los recursos. Con esta herramienta hemos utilizado una máquina de 4 núcleos y 23 GB de RAM.
- **BigQuery**: Es un almacén de datos empresariales de Google de gran escalabilidad y sin servidor diseñado para mejorar la productividad de los analistas de datos con un precio y un rendimiento inigualables. Como no hay que gestionar ninguna infraestructura, puedes centrarte en analizar los datos para encontrar información importante mediante el conocido lenguaje SQL y sin necesidad de un administrador de bases de datos.
- **Storage**: Sistema de almacenamiento de objetos, que permite archivar datos no estructurados y ficheros de gran tamaño (PB), autogestionables y fácilmente integrables con el resto de servicios de la Google Cloud Platform. Lo hemos utilizado para el almacenamiento de los .csv con datos de entrenamiento y test.
- **Datalab**: Herramienta interactiva de gran potencia para explorar datos, analizarlos y visualizarlos, así como desarrollar modelos de aprendizaje automático en la Cloud Platform. Se ejecuta en la Compute Engine y se conecta fácilmente al resto de servicios de la Cloud sin necesidad de autenticación para poder centrarse en las tareas relacionadas con la explotación de datos. Es similar a un notebook de Jupyter pero con mejor integración en este ecosistema.

1.4 Estructura de la Memoria

La memoria está estructurada en 5 capítulos en los que abordamos diferentes pasos que hemos realizado durante la realización de este Trabajo de Fin de Grado:

En el primer capítulo hemos realizado una introducción a la herramienta Google Analytics, familiarizándonos así con su funcionamiento y los datos que nos provee. También hemos definido los objetivos a alcanzar y la motivación de estos.

En el segundo capítulo estudiamos los fundamentos del aprendizaje supervisado. Hacemos una introducción histórica que nos ayuda a comprender el estado del arte, sus hitos y su potencial. Después, se describe su funcionamiento de forma teórica para comprender cómo funcionan los diferentes modelos y por qué elegir esos mismos. Además, se mencionan ciertas técnicas que, en conjunción con los modelos seleccionados, pueden ayudar a mejorar los resultados de estos.

En el tercer capítulo se realiza un exhaustivo análisis de los datos con el fin de ayudar a tomar ciertas decisiones sobre estos. Estas decisiones resultan cruciales para el resultado del Trabajo de Fin de Grado. En primer lugar, se realiza un análisis por columnas, centrándonos en el significado de las mismas. En segundo lugar, se realiza un análisis por filas, observando la relación entre las

diferentes columnas en un mismo registro. Finalmente, se crean una serie de características que pueden ayudar en las predicciones del modelo.

El cuarto capítulo entra en la parte práctica del aprendizaje automático, implementando diferentes modelos, comparado estos y eligiendo el más adecuado.

Para acabar, en el quinto capítulo, mostramos las conclusiones a las que hemos llegado durante el desarrollo del Trabajo de Fin de Grado y las líneas futuras a seguir para trabajos sucesivos complementarios a éste.

Además, incluimos un anexo en el que explicamos los cursos realizados como preparativo para este Trabajo de Fin de Grado. Estos cursos pueden resultar de gran interés a modo de introducción al mundo del aprendizaje automático.

Capítulo 2. Aprendizaje Supervisado

El aprendizaje supervisado es la técnica que seguir en este Trabajo de Fin de Grado. En este capítulo, desglosaremos distintas opciones de diseño, mostrando fundamentos teóricos, analizados previamente al comienzo de la toma de decisiones.

2.1 Introducción

Para entender nuestro Trabajo de Fin de Grado sobre la predicción del beneficio por cliente basándonos en datos de Google Analytics con técnicas de Machine Learning no podría entenderse sin una explicación sobre la Inteligencia Artificial (AI, *Artificial Intelligence*) y el Machine Learning.

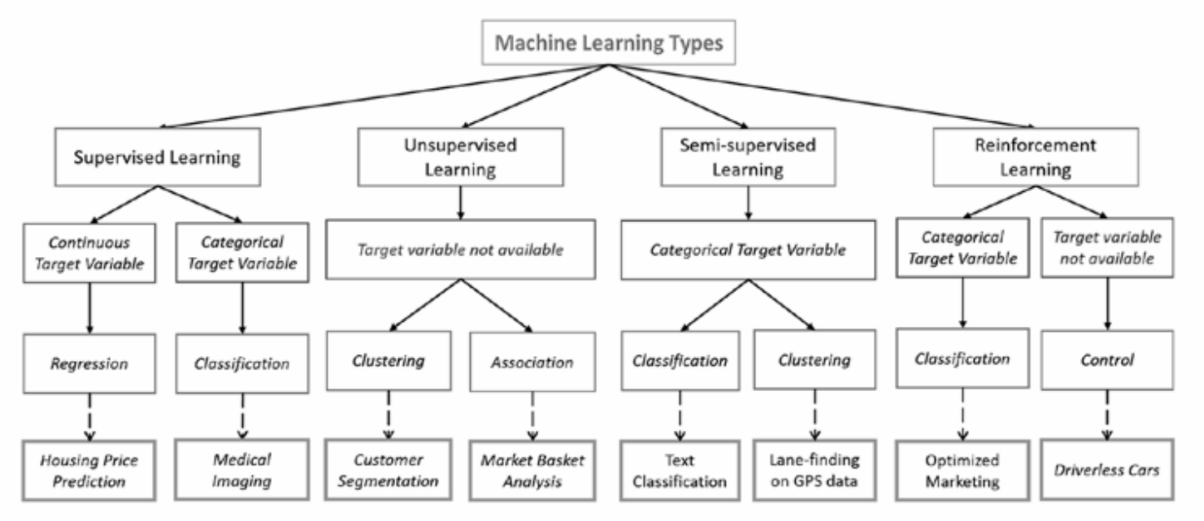


Ilustración 2.1 Esquema de diferentes tipos de Aprendizaje Automático

En la Ilustración 2.1 podemos ver cómo el Aprendizaje Supervisado es una rama del Machine Learning. Debido a esto, es imprescindible hablar de Inteligencia Artificial y Machine Learning para después derivar en el Aprendizaje Supervisado.

La Inteligencia Artificial se conoce como el estudio y diseño de agentes inteligentes. Un agente es algo que actúa en un entorno. Un agente inteligente es un sistema que actúa de forma inteligente, es decir, que actúa de forma adecuada a las circunstancias y su objetivo, es flexible ante cambios en su entorno y un cambio de objetivo, aprende de la experiencia y realiza cambios apropiados con una percepción y capacidad de computación limitada.

El diseño de estos agentes implica el poder desarrollar modelos matemáticos complejos sin necesidad de programar una infinidad de reglas que abarque las combinaciones posibles para abarcar nuestro problema.

En los primeros días de la inteligencia artificial, este campo abordó y resolvió rápidamente problemas que son intelectualmente difíciles, pero relativamente sencillos para las computadoras, problemas que pueden ser fácilmente descritos mediante una lista de reglas formales matemáticas. Sin embargo, el verdadero desafío al que se enfrenta la Inteligencia Artificial es resolver tareas que son relativamente fáciles para una persona, pero enormemente complejos para una computadora,

debido a la dificultad de describirlos formalmente. Estos problemas son aquellos que las personas realizamos intuitivamente, de forma automática, por ejemplo, cómo reconocer palabras o caras en una imagen (Goodfellow, I. et al., 2016).

En el presente, la Inteligencia Artificial es un campo próspero con muchas aplicaciones prácticas temas de investigación en activo. Desarrollamos este software inteligente para automatizar el trabajo de rutina, entender el habla o las imágenes, hacer diagnósticos en medicina y apoyar la investigación científica básica (Goodfellow, I. et al., 2016).

Las dificultades a las que se enfrentan estos sistemas sugieren que los sistemas de inteligencia artificial necesitan la capacidad de adquirir su propio conocimiento mediante la extracción de patrones a partir de datos. Esta capacidad se conoce como Machine Learning. La introducción del Machine Learning permitió a los ordenadores el resolver problemas del mundo real, tomando decisiones aparentemente subjetivas. Un simple algoritmo de Machine Learning puede decidir si recomendar o no la realización de una cesárea (Mor-Yosef et al. 1990, citado en Goodfellow, I. et al. 2016).

El Machine Learning es un subcampo de la ciencia de computación que se ocupa de construir algoritmos que, para ser útiles, se basan en una colección de ejemplos de un fenómeno determinado. Estos ejemplos pueden provenir de la naturaleza, ser creados por humanos o generados por otro algoritmo. El Machine Learning también se puede definir como el proceso de resolver un problema práctico de recopilar un conjunto de datos y construir algorítmicamente un modelo estadístico basado en ese conjunto de datos. Se supone que ese modelo estadístico se utiliza de alguna manera para resolver el problema práctico (Burkov, A. *et* al. 2019).

Hay cuatro tipos principales de aprendizaje automático: aprendizaje supervisado, aprendizaje no supervisado, aprendizaje semisupervisado y aprendizaje por refuerzo. La diferencia entre estos cuatro tipos radica en la forma de los datos de entrenamiento, así como en la técnica empleada para desarrollar el modelo. En este Trabajo de Fin de Grado nos hemos centrado en técnicas de aprendizaje supervisado.

2.2 Funcionamiento del Aprendizaje Supervisado

En aprendizaje supervisado, el set de datos es una colección de ejemplos etiquetados $\{(\mathbf{x}_i, y_i)\}^N_{i=1}$. cada elemento $\mathbf{x}_i$ es llamado vector de características. Un vector de características es un vector el cual en cada dimensión $j = 1, ..., D$ contiene un valor que describe el ejemplo de alguna forma. Ese valor, se llama característica y es denotado como $x^{(j)}$. Para todos los ejemplos del set de datos, la característica en la posición j contiene el mismo tipo de información. La etiqueta y_i puede ser tanto un elemento perteneciente a un set finito de clases $\{1, 2, \ldots, C\}$, un número real o incluso una estructura más compleja como un vector, una matriz, un árbol o un gráfico. Podemos ver una clase como una categoría a la que el ejemplo pertenece.

El objetivo de un algoritmo de aprendizaje supervisado es usar el set de datos para producir un modelo que, a partir de un vector $\mathbf{x}_i$ como entrada, consiga deducir una etiqueta y_i para este vector de características.

2.3 Teoría

En este apartado vamos a ver la teoría relacionada con lo que hemos trabajado en este Trabajo de Fin de Grado. Comenzaremos con los términos comunes a los modelos y finalizamos con una explicación de cada uno de los cuales hemos utilizado.

Los modelos de Machine Learning son sistemas que comparten muchas características comunes. Incluso si, a veces, estos modelos han sido definidos desde un punto de vista teórico, los avances en investigación nos permiten entender el comportamiento de sistemas complejos como las redes neuronales profundas (Bonaccorso, G. et al 2018).

Los algoritmos de Machine Learning trabajan con datos. Estos algoritmos crean asociaciones, buscan relaciones, descubren patrones, generan nuevas muestras y mucho más, trabajando con sets de datos bien definidos. Desafortunadamente, a veces las suposiciones o las condiciones impuestas en ellos no son claras, y un largo proceso de entrenamiento puede acabar en un estrepitoso fracaso. Podemos pensar en un modelo como una caja gris, donde una entrada matricial **X** se transforma en una salida vectorial **y** (Bonaccorso, G. et al 2018).

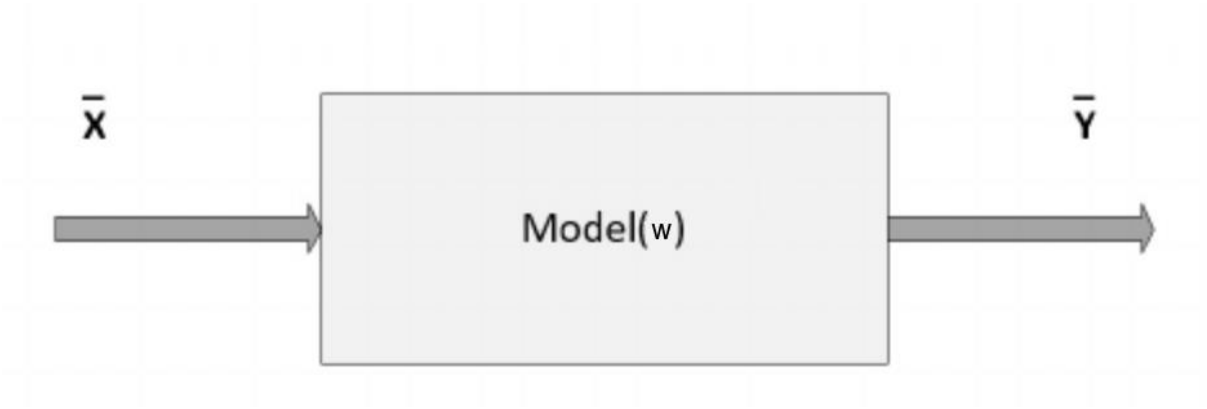


Ilustración 2.2 Representación de modelo genérico (Bonaccorso, G. et al 2018)

En la Ilustración 2.2, el modelo ha sido representado por una pseudo-función que depende de un set de parámetros definidos por el vector **w**. Por ahora solo consideraremos los modelos paramétricos, dado que se basan solo en la estructura de los datos.

La tarea del proceso paramétrico de aprendizaje es encontrar la mejor combinación de parámetros que maximiza una función objetivo cuyo valor es proporcional a la precisión del modelo, dadas una entrada **X** y una salida **y**. Aunque esta definición no es muy rigurosa, es muy útil para entender el contexto en el que trabajamos (Bonaccorso, G. et al 2018).

Para los problemas de aprendizaje supervisado siempre tendremos. Una entrada **X**, la función objetivo, desconocida para nosotros, $f: X \rightarrow Y$, donde X es el espacio de las entradas (set de todas las entradas posibles **X**) e Y es el espacio de salidas (set de todas las salidas posibles **y**). Hay un set de datos D para calcular la fórmula $g: X \rightarrow Y$ que aproxima f. El algoritmo elige g de un set de fórmulas candidatas bajo consideración, el cual llamamos set de hipótesis H.

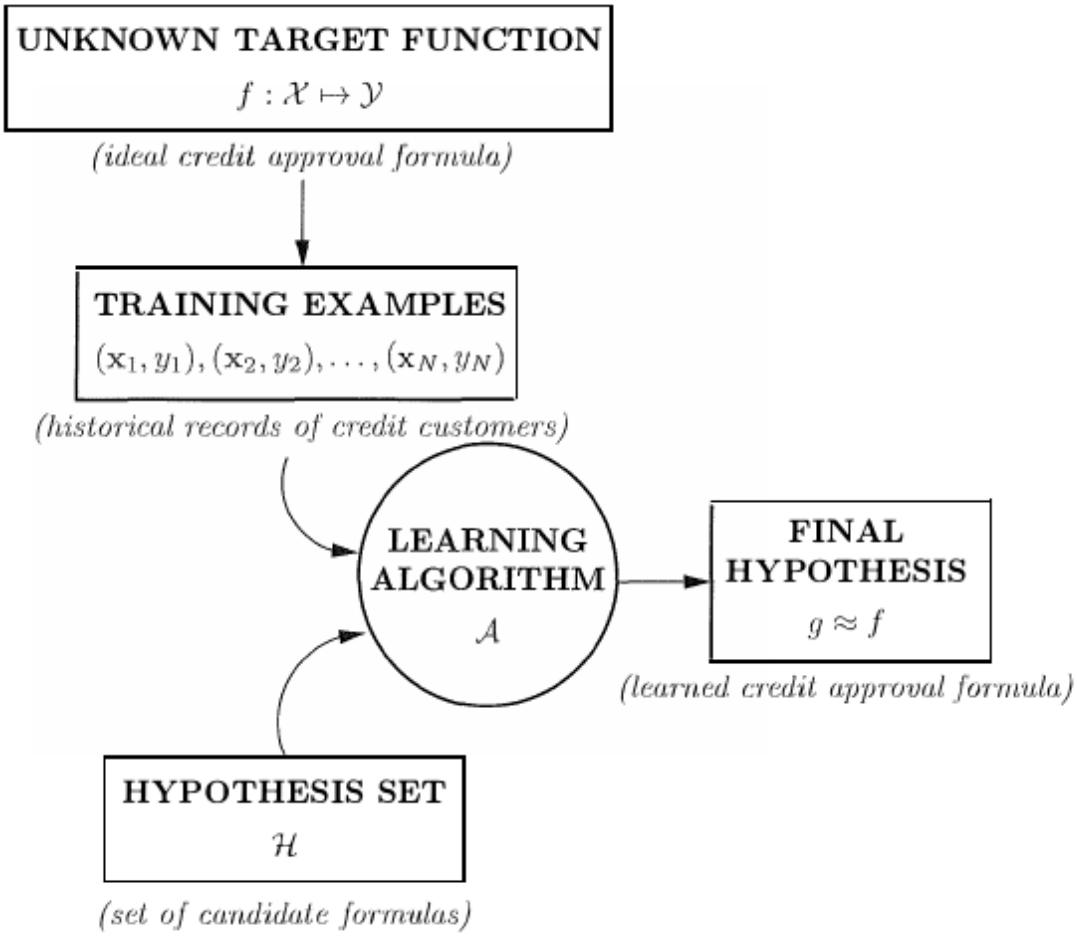


Ilustración 2.3 Diagrama de hipótesis final (Abu-Mostafa, Y. S. et al 2012)

Como vemos en la Ilustración 2.3, el algoritmo elige la *g* que mejor corresponda con *f* en el set de entrenamiento, con la esperanza de que encaje de la misma forma con nuevas muestras (Abu-Mostafa, Y. S. et al 2012).

2.3.1 Ejemplo con regresión lineal

La regresión lineal es el modelo más sencillo utilizado en el aprendizaje supervisado. Su función se trata de una combinación lineal de las características del ejemplo de entrada.

Teniendo una colección de ejemplos etiquetados, $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ donde *N* es el tamaño de la colección, $\mathbf{x}_i$ de dimensión *D* es el vector de características del ejemplo $i = 1, ..., N$, y_i es la etiqueta de valor real y cada característica $x_i^{(j)}$, $j = 1, ..., D$ es también un número real.

Queremos crear un modelo $f_{\mathbf{w},b}(\mathbf{x})$ como combinación lineal de características del ejemplo **x**:

$f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}\mathbf{x} + b$

donde **w** es un vector de dimensión *D* de parámetros y *b* es un número real. La notación $f_{\mathbf{w},b}$ significa que el modelo *f* está parametrizado por dos valores **w** y *b*.

Vamos a usar el modelo para predecir la y para un dado **x** tal que $y \leftarrow f_{w,b}(\mathbf{x})$. Buscamos encontrar los valores óptimos (**w***, *b**) que definen el modelo que realiza predicciones más precisas.

Para satisfacer este último requerimiento, el proceso de optimización que suele utilizarse para encontrar los valores óptimos **w*** y *b** pasa por minimizar la siguiente expresión:

$$\frac{1}{N}\sum_{i=1\ldots N}(f_{\mathbf{w},b}(\mathbf{x}_i)-y_i)^2.$$

La expresión que minimizamos o maximizamos se llama función objetivo. La expresión $(f(\mathbf{x}_i) - y_i)^2$ es la función de pérdida. La función de pérdida es una medida de penalización por una mala clasificación del ejemplo *i*. Esta función de pérdida en particular se llama error cuadrático medio. Todos los algoritmos basados en modelos tienen una función de pérdida y lo que se hace para encontrar el mejor modelo es minimizar la función objetivo, también conocida como función de coste. En una regresión lineal, la función de coste viene dada por la pérdida promedio, también llamada riesgo empírico. La pérdida promedio o riesgo empírico, para un modelo, es la media de las penalizaciones obtenidas aplicando el modelo al set de entrenamiento (Burkov, A. et al 2019).

El problema de estas funciones de pérdida es que miden cuánto se adapta la función al set de datos utilizados, pero no debemos perder nuestro objetivo, que es encontrar una función *f* objetivo válida en casos generales. Por esto existen dos medidas de error diferentes:

- E_{in}: Error que se comete en la muestra de datos que tenemos (nuestro set de entrenamiento).
- E_{out}: Error que se comete en la población completa (fuera del set de entrenamiento).

El objetivo es minimizar el error E_{out} con el fin de que nuestra función acierte en la población completa. Sin embargo, no podemos hallar el E_{out} de nuestra función *g* al no tener todas las muestras de la población, solo podemos estimarlo.

Al contrario que el E_{out}, el error de nuestra muestra E_{in} sí es calculable. Dado un conjunto de *N* datos, un modelo complejo reduce E_{in}, pero, al aumentar la complejidad del modelo, estamos aumentando el error de generalización, con otras palabras, estamos adaptando nuestra función *g* exclusivamente al set de datos que tenemos.

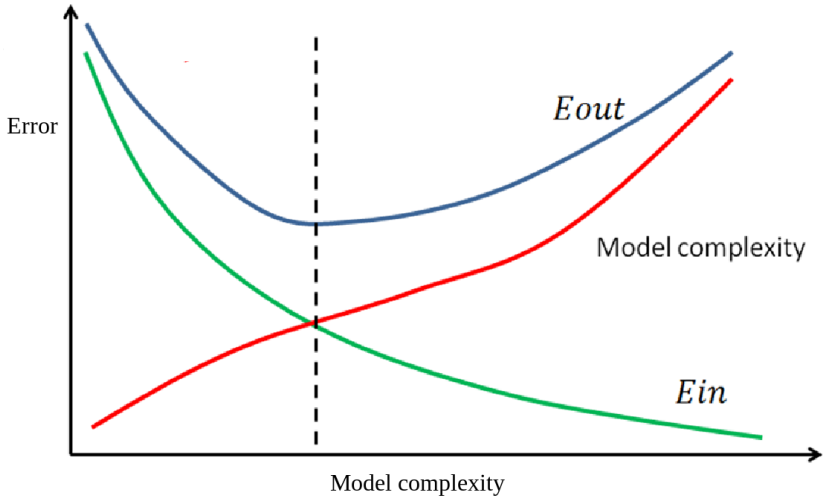


Ilustración 2.4 Punto óptimo del modelo

En la Ilustración 2.4 podemos ver cómo, para un conjunto de *N* datos, al aumentar la complejidad del modelo, el E_{in} tiende a 0, mientras que el E_{out} no para de crecer. La línea vertical de puntos describe el punto óptimo, donde el compromiso de la complejidad del modelo obtiene el mejor resultado, es decir, el valor de E_{out} mínimo.

Cuando nos situamos a la izquierda del punto óptimo, se dice que hay *underfitting*, mientras que cuando nos situamos a la derecha se denomina *overfitting*.

- **Underfitting:** Si se utiliza un modelo demasiado simple, no se ajusta bien a los datos, por lo que se produce un problema de sesgo o de *underfitting*. Como vemos en la Ilustración 2.5, si utilizamos un clasificador lineal simple en el set de datos mostrado, este clasificador no podrá separar correctamente las dos clases, quedando así un modelo con *underfitting*. El mayor indicador del *underfitting* es un E_{in} muy alto, que se corresponde a una baja precisión de entrenamiento. La única solución posible será adoptar un modelo de mayor complejidad.

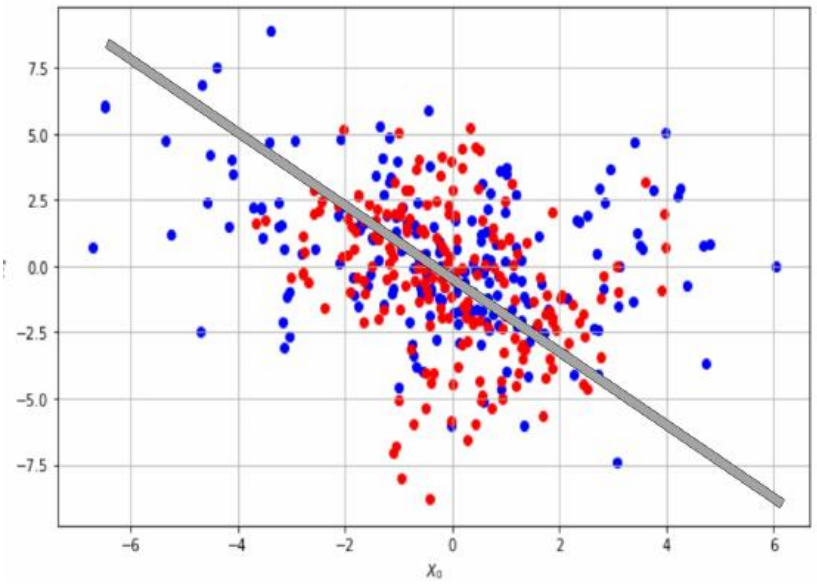


Ilustración 2.5 Regresión lineal con datos no linealmente separables

- **Overfitting:** El *overfitting* se produce cuando el modelo es demasiado complejo para representar la función objetivo teniendo en cuenta la cantidad de datos disponibles para entrenar ese modelo. El modelo utiliza sus grados adicionales de libertad para ajustar las idiosincrasias en los datos, lo que arroja una hipótesis final que es inexacta: se ajusta muy bien a los datos (E_{in} muy bajo) pero no generaliza bien por lo que da lugar a un E_{out} elevado. El *overfitting* puede ocurrir incluso cuando el conjunto de hipótesis sólo contiene funciones más simples que la función objetivo (Abu-Mostafa, Y. S. et al 2012).
  Hay varias soluciones posibles para el problema del *overfitting*:
  - Usar un modelo más simple.
  - Reducir la dimensionalidad de los ejemplos del set de datos.
  - Añadir más datos al set de entrenamiento (no siempre es posible).
  - Regularizar el modelo.

La regresión lineal es un modelo sencillo (comparado con otras opciones). El utilizar el error cuadrático medio como función de pérdida se debe a que su derivada es continua, al contrario que la derivada del valor absoluto de la diferencia. Al no tener una derivada continua, se dice que la función no es suave. Las funciones no suaves crean dificultades innecesarias cuando aplicamos álgebra lineal para problemas de optimización. Las soluciones cerradas para encontrar el óptimo de una función suelen ser más fáciles de implementar que los complejos métodos numéricos de optimización, como el descenso de gradiente.

Si no es posible añadir datos al set de entrenamiento, el aumento de datos (*data augmentation*) puede ser una solución válida, porque permite crear muestras artificiales (ej: en imágenes, rotarlas, hacer espejo, o difuminarlas) empezando por la información que guardan las ya conocidas (Bonaccorso, G. et al 2018).

La regularización es el método más utilizado para prevenir el *overfitting* (Burkov, A. et al 2019), junto con el uso de las técnicas de validación (Abu-Mostafa, Y. S. et al 2012).

2.3.1.1 Regularización

Incluso el modelo más simple puede sobreajustarse a los datos. Esto normalmente pasa cuando los datos tienen muchas dimensiones, pero el set de entrenamiento tiene pocos ejemplos. De hecho, cuando los vectores de características tienen muchas dimensiones, el algoritmo de aprendizaje lineal puede crear un modelo que asigna valores no nulos a la mayoría de las dimensiones $w^{(j)}$ en el vector de parámetros **w**, intentando encontrar relaciones demasiado complejas entre las características disponibles para predecir las etiquetas de los ejemplos de entrenamiento de forma perfecta (Burkov, A. *et* al. 2019).

La regularización es un arma para combatir el *overfitting*. Ésta restringe el algoritmo de aprendizaje para mejorar el E_{out}, especialmente cuando el ruido está presente. Podemos ver en la Ilustración 2.6 mostrada a continuación, como una pequeña regularización mejora dramáticamente los resultados (Abu-Mostafa, Y.S. et al 2012).

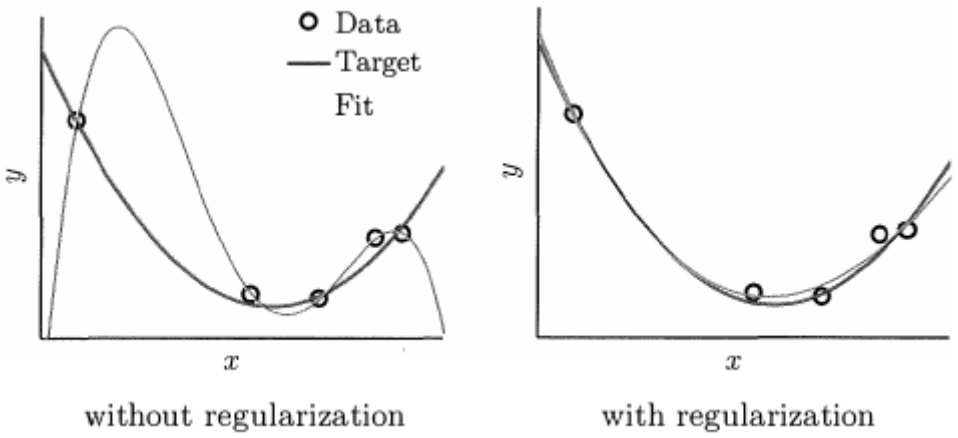


Ilustración 2.6 Comparativa de modelo con y sin regularización

Los dos tipos de regularización más conocidos, en el caso de la regresión lineal, se llaman regularización L1 o regularización Lasso y regularización L2 o regularización Ridge. Para crear un modelo regularizado, modificamos la función objetivo añadiendo un término de penalización cuyo valor es mayor cuanto más complejo es el modelo (Burkov, A. *et al* 2019).

- **Regularización Lasso (L1):** En la regularización Lasso, la complejidad *C* se mide como

$$C = \frac{1}{N}\sum_{j=1}^{N}|w_i|$$

  la media del valor absoluto de los coeficientes del modelo. Es un tipo de regularización muy útil cuando sospechamos que algunos de los atributos de nuestro vector de características pueden ser irrelevante. Al usar la regularización L1, estamos fomentando una menor densidad de la solución, favoreciendo que los coeficientes puedan acabar valiendo 0. Esta regularización puede ayudar cuando los atributos no están muy correlados entre ellos.

- **Regularización Ridge (L2):** En la regularización Ridge, la complejidad *C* se mide como

$$C = \frac{1}{2N}\sum_{j=1}^{N}w_i^2$$

  la media del cuadrado de los coeficientes del modelo. La regularización L2 hace que los pesos de los coeficientes sean pequeños, generalizando mejor.

- **Regularización ElasticNet:** ElasticNet combina ambas regularizaciones (L1 y L2) con un parámetro *r* que indica la importancia relativa de cada una:

$$C = r \cdot Lasso + (1 - r) \cdot Ridge$$

Es efectiva cuando disponemos de muchos atributos en nuestro vector de características, estando algunos muy correlados, pero siendo otros muy poco útiles.

2.3.1.2 Validación

Con este método, mantendremos una parte de los datos como conjunto de datos de prueba. En este proceso, entrenamos el modelo ajustando sus parámetros con los datos de entrenamiento y, finalmente, evaluando con los datos restantes, a los que llamaremos datos de validación.

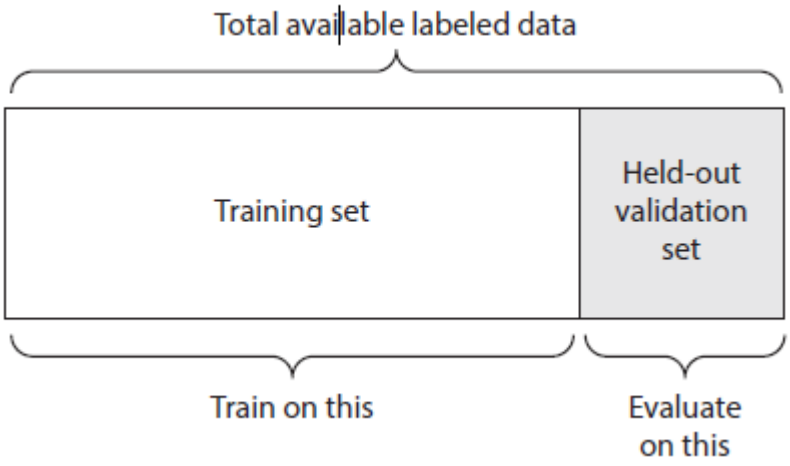


Ilustración 2.7 Separación de datos en sets de entrenamiento y validación

De esta manera tendríamos, un set de datos de entrenamiento, un set de datos de validación. De esta manera, puede surgir un inconveniente cuando disponemos de una cantidad limitada de datos y es que la cantidad de los conjuntos de entrenamiento y validación pueden ser tan pequeños que no conseguiremos un modelo efectivo. Para evitar este problema, podemos utilizar la validación cruzada, de la cual podemos utilizar dos tipos principalmente.

- ***Leave-One-Out* (LOOCV):** Dispondremos de tantas iteraciones como ejemplos dispongamos en el set de entrenamiento. En cada iteración se entrena con todos los ejemplos menos 1, al que aplicaremos la hipótesis y sobre el que calcularemos el error obtenido. El error de validación LOOCV se calcula con la media de los errores obtenidos en cada ejemplo.
- ***K-Fold*:** Los datos se dividen en K pliegues y hay K iteraciones. En cada una se entrena con K-1 pliegues y se aplica la hipótesis al pliegue restante y se calcula el error cometido en ese pliegue. El error de validación cruzada *K-Fold* se calcula con la media del error cometido en cada pliegue.

Una vez obtenidos estos errores, podremos elegir un modelo de entre varios de acuerdo con el error de validación cruzada que hayamos obtenido con cada uno. Después de elegir el modelo, entrenaremos con todos los datos disponibles antes de utilizarlo (tanto los del set de entrenamiento como los del set de validación). El inconveniente principal de esta técnica es que tiene sesgo de selección. Sin embargo, este sesgo de selección no es significativo para sets de datos moderadamente grandes (Kuhn, M. *et al* 2016) o si no probamos demasiados modelos de aprendizaje (Abu-Mostafa, Y. S. et al 2012).

2.3.1.3 Normalización

Otra técnica que, aunque no está diseñada para combatir el *overfitting*, generalmente ayuda a mejorar los modelos es la normalización. Con la normalización se trata de aplicar una serie de transformaciones a los datos para poder sacar el máximo partido a los modelos. Las dos principales técnicas son:

- **Centrado en cero:** Muchos algoritmos obtienen un mejor rendimiento cuando el set de datos es simétrico. Debido a esto, uno de los pasos más importantes de preprocesado es el centrado en cero, que consiste en extraer la media $E_x[X]$ de todas las muestras:

$$\hat{x}_i = \bar{x}_i - E_x[X]$$

  Esta operación, si es necesaria, es reversible y no altera las relaciones entre las muestras ni la cantidad de componentes en la misma muestra. En algunos escenarios, un set de datos centrado en cero permite explotar la simetría de algunas funciones de activación, mejorando significativamente la convergencia.

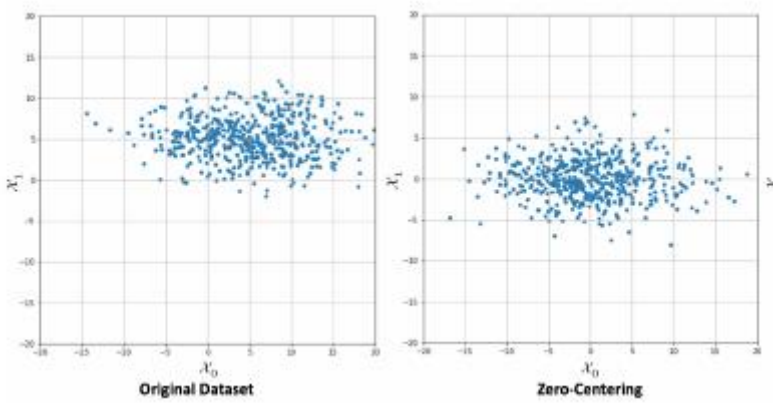


Ilustración 2.8 Diferencia entre set de datos original y set de datos centrado en cero

- **Escalado:** En el caso del escalado, se normalizan las entradas entre unos límites definidos. Hay diferentes tipos de escalado, los más conocidos son (Towards Data Science, 2019):
  - Escalado estándar: Estandariza las características mediante la supresión de la media y la escala a varianza unidad.

$$Z = (X - E(X))/\sigma^2(x)$$

  - Escalado por máximo y mínimo: Escala las características según sus valores máximos y mínimos.

$$Z = (X - \min(X))/(\max(X) - \min(X))$$

2.3.2 Otros modelos interesantes

Hablaremos ahora de otro tipo de modelos más complejos que vamos a utilizar en la parte práctica de este Trabajo de Fin de Grado.

2.3.2.1 Árbol de decisión

Un árbol de decisión es un gráfico acíclico que puede ser usado para tomar decisiones. En cada rama del gráfico se examina una característica *j* del vector de características. Si el valor de la característica está debajo de un umbral (en caso de ser numérico), entonces se sigue una de las ramas, sin embargo, si está por encima, se sigue la otra.

Los árboles de decisión pueden utilizarse tanto para un problema de regresión como para uno de clasificación. Nosotros nos centraremos en los árboles de regresión, dado que son los que interesan para el problema expuesto en este Trabajo de Fin de Grado. Los árboles de decisión más ampliamente utilizados para regresión siguen el algoritmo CART (por las siglas en inglés de Árboles de Clasificación y Regresión).

La representación del modelo CART es un árbol binario. Cada nodo raíz representa una única variable de entrada $x^{(j)}$ y un punto de división en esa variable (asumiendo que es numérica). Los nodos de cada hoja del árbol contienen una variable de salida **y** que se utiliza para hacer una predicción (Machine Learning Mastery, 2019).

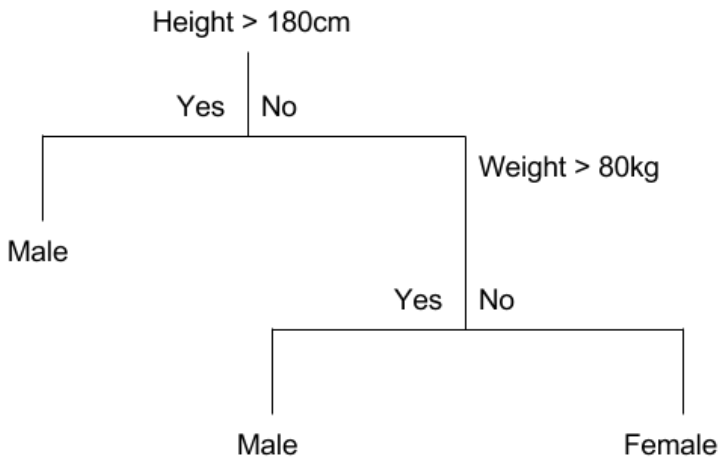


Ilustración 2.9 Ejemplo sencillo de árbol de decisión (Machine Learning Mastery, 2017).

En la Ilustración 2.9 podemos ver un ejemplo con un conjunto de datos con dos entradas $x^{(1)}$ y $x^{(2)}$ que corresponderían a la altura en centímetros y el peso en kilogramos, la salida sería el sexo como hombre o mujer.

Con la representación en árbol binario del modelo CART descrito anteriormente, hacer predicciones es relativamente sencillo. Dada una nueva entrada, el árbol se recorre al evaluar la entrada específica iniciada en el nodo raíz del árbol.

Un árbol binario es en realidad una partición del espacio de entrada. Puede pensar en cada variable de entrada como una dimensión en un espacio p-dimensional. El árbol de decisión lo divide en rectángulos (cuando p = 2 variables de entrada) o en algún tipo de hiperrectángulo con más entradas.

Los nuevos datos se filtran a través del árbol y aterrizan en uno de los rectángulos y el valor de salida para ese rectángulo es la predicción hecha por el modelo. Esto da una idea del tipo de decisiones que un modelo de CART es capaz de tomar, por ejemplo, límites de decisiones en forma cuadrada.

Crear un modelo CART implica seleccionar variables de entrada y puntos de división en esas variables hasta que se construya un árbol adecuado. La selección de la variable de entrada que se debe utilizar y la división específica o punto de corte se elige mediante un algoritmo codicioso para minimizar una función de costo. La construcción del árbol finaliza utilizando un criterio de detención predefinido, como un número mínimo de instancias de entrenamiento asignadas a cada nodo de hoja del árbol.

La creación de estos árboles se realiza teniendo en cuenta una serie de pasos:

- **División "codiciosa":** Crear un árbol de decisiones binario es en realidad un proceso de división del espacio de entrada. Se utiliza un enfoque codicioso para dividir el espacio denominado división binaria recursiva. Este es un procedimiento numérico en el que todos los valores están alineados y se prueban diferentes puntos de división utilizando una función de coste. Se selecciona la división con el coste más bajo.

  Todas las variables de entrada y todos los puntos de división posibles se evalúan y eligen de manera codiciosa (por ejemplo, el mejor punto de división se elige cada vez).

- **Criterio de parada:** El procedimiento de división binaria recursiva descrito anteriormente necesita saber cuándo dejar de dividir a medida que avanza hacia abajo en el árbol con los datos de entrenamiento.

  El procedimiento de detención más común es utilizar un recuento mínimo en el número de instancias de entrenamiento asignadas a cada nodo hoja. Si el recuento es menor que algún mínimo, la división no se acepta y el nodo se toma como un nodo de hoja final.

- **Poda del árbol:** El criterio de detención es importante ya que influye fuertemente en el rendimiento de su árbol. Puede usar la poda después de haber aprendido el árbol para aumentar aún más el rendimiento y generalizar mejor.

  La complejidad de un árbol de decisión se define como el número de divisiones en el árbol. Suelen preferirse árboles más simples. Son más fáciles de entender, y generalizan mejor.

  El método de poda más rápido y simple es trabajar a través de cada nodo de hoja en el árbol y evaluar el efecto de eliminarlo utilizando un conjunto de prueba de retención. Los nodos de la hoja se eliminan solo si se produce una caída en la función de coste total en todo el conjunto de pruebas. Se dejan de eliminar nodos cuando no se pueden realizar más mejoras.

Se pueden usar métodos de poda más sofisticados, como la poda de complejidad de costes (también llamada poda de enlace más débil) donde se usa un parámetro de aprendizaje (alfa) para evaluar si los nodos se pueden eliminar según el tamaño del subárbol.

2.3.2.2 Máquinas de vectores de soporte

Las máquinas de vectores de soporte (SVM) estuvieron en sus inicios enfocadas en resolver problemas de clasificación binaria, sin embargo, actualmente se utilizan para resolver otros tipos de problemas como, en el caso de nuestro Trabajo de Fin de Grado, la regresión.

En estos casos en los que utilizamos las SVM para regresión, suelen denominarse por SVR. Así, dado un conjunto de ejemplos de entrenamiento $S = \{(\mathbf{x_1}, y_1), \ldots.(\mathbf{x_n}, y_n)\}$, donde $\mathbf{x_i} \in \mathbb{R}^d$ e $y_i \in \mathbb{R}$, en el que se asume que los valores y_i de todos los ejemplos de S se pueden ajustar mediante una función lineal, el objetivo de la tarea de regresión es encontrar los parámetros $\mathbf{w} = (w_1, \ldots, w_d)$ que permitan definir dicha función lineal .

$f(\mathbf{x}) = (w_1 x_1 + \ldots w_d x_d) + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$

Para permitir cierto ruido en los ejemplos de entrenamiento se pueden relajar la condición de error entre el valor predicho por la función y el valor real. Para ello, se utiliza la denominada *función de pérdida* ϵ-*insensible, L*ϵ, caracterizada por ser una función lineal con una zona insensible de anchura 2ϵ, en la que el error es nulo.

$$L_\epsilon(y, f(\boldsymbol{x})) = \begin{cases} 0 & \text{si } |y - f(\boldsymbol{x})| \leq \epsilon \\ |y - f(\boldsymbol{x})| - \epsilon & \text{en otro caso} \end{cases}$$

Dado que en la práctica es muy difícil que los ejemplos de entrenamiento se ajusten al modelo lineal con un error de predicción igual a cero, se recurre al concepto de margen blando. Para ello se definen dos variables de holgura ξ^+_i y ξ^-_i, que permitirán cuantificar la magnitud de dicho error. Así, la variable $\xi^+_i > 0$ cuando la predicción del ejemplo, $f(\mathbf{x_i})$ es mayor que su valor real, y_i, en una cantidad superior a ϵ, es decir $f(\mathbf{x_i}) - y_i > \epsilon$. En otro caso, su valor será 0. De forma similar, la variable $\xi^-_i > 0$ cuando la predicción del ejemplo, $f(\mathbf{x_i})$ es mayor que su valor real, y_i, en una cantidad superior a ϵ, es decir $y_i - f(\mathbf{x_i}) > \epsilon$. En otro caso, su valor será 0.

Dado que no puede ocurrir simultáneamente que la predicción de un ejemplo sea a la vez ($\xi^+_i > 0$) y menor ($\xi^-_i < 0$) que su valor real, se puede afirmar que $\xi^+_i \cdot \xi^-_i = 0$ (Carmona, E. *et. als.* 2014).

De esta forma, la suma de todas las variables de holgura permitirá, de alguna manera, medir el coste asociado al número de ejemplos con un error de predicción no nulo. Por tanto, la función a optimizar será:

$$f(\boldsymbol{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

Con la salvedad de que tenemos dos tipos de variables de holgura. Luego el problema queda definido como:

$$\begin{aligned} \text{mín} \quad & \tfrac{1}{2} < \boldsymbol{w}, \boldsymbol{w} > + C \sum_{i=1}^{n} \left(\xi_i^+ + \xi_i^-\right) \\ \text{s.a.} \quad & (< \boldsymbol{w}, \boldsymbol{x}_i > + b) - y_i - \epsilon - \xi_i^+ \leq 0 \\ & y_i - (< \boldsymbol{w}, \boldsymbol{x}_i > + b) - \epsilon - \xi_i^- \leq 0 \\ & \xi_i^+, \xi_i^- \geq 0, \ i = 1, \ldots, n \end{aligned}$$

2.3.2.3 Red Neuronal Prealimentada (*Feedforward neural network*)

Las redes neuronales prealimentadas (*feedforward neural networks*) también tienen como objetivo aproximar una función *f* y para ello están formadas por varias capas, cada una de ellas con un cierto número de unidades o neuronas. Estos modelos son llamados "prealimentados" porque la información fluye a través de las distintas capas desde la entrada a la salida. Cuando una red neuronal incluye conexiones de retroalimentación son llamadas redes neuronales recurrentes (I. Goodfellow et als 2019).

Durante el entrenamiento de la red neuronal, manejamos $f(\mathbf{x})$ para que coincida con $f(\mathbf{x})$. Los datos de entrenamiento nos proporcionan ejemplos ruidosos y aproximados de $f(\mathbf{x})$ evaluados en diferentes puntos de entrenamiento. Cada ejemplo **x** va acompañado de una etiqueta $y \approx f(\mathbf{x})$. Los ejemplos de entrenamiento especifican directamente lo que debe hacer la capa de salida en cada punto, debe producir un valor cercano a y. El comportamiento de las otras capas no está directamente especificado por los datos de entrenamiento. El algoritmo de aprendizaje debe decidir cómo usar esas capas para producir la salida deseada, pero los datos de entrenamiento no dicen qué debe hacer cada capa individual. En cambio, el algoritmo de aprendizaje debe decidir cómo usar estas capas para implementar mejor una aproximación de $f(\mathbf{x})$. Debido a que los datos de entrenamiento no muestran el resultado deseado para cada una de estas capas, estas capas se denominan capas ocultas.

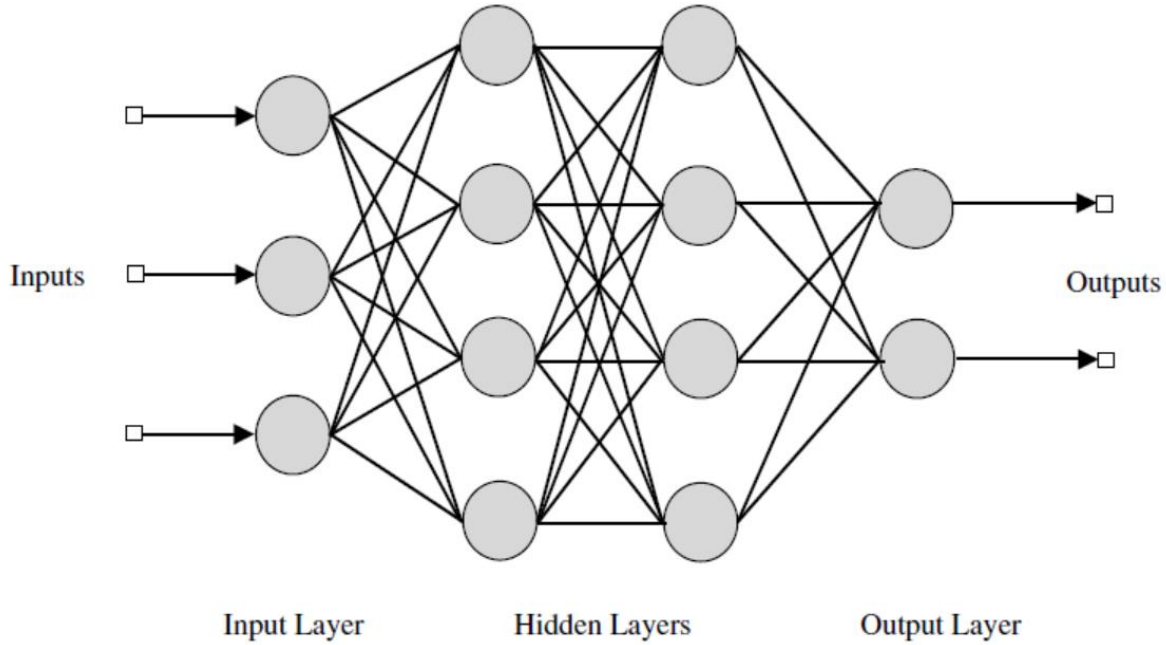


Ilustración 2.10 Red Neuronal Prealimentada

Estas redes se denominan neurales porque están vagamente inspiradas en su homólogo biológico. Cada capa oculta de la red suele tener un valor vectorial. La dimensionalidad de estas capas ocultas determina el ancho del modelo. Cada elemento del vector puede interpretarse como un papel análogo a una neurona. En lugar de pensar que la capa representa una sola función de vector a vector, también podemos pensar que la capa consiste en muchas unidades que actúan en paralelo, cada una representando una función de vector a escalar. Cada unidad se asemeja a una neurona en el sentido de que recibe información de muchas otras unidades y calcula su propio valor de activación. La idea de usar muchas capas de representación con valores vectoriales se extrae de la neurociencia. La elección de las funciones $f_i^*(\mathbf{x})$ utilizadas para calcular estas representaciones también se guía libremente por observaciones neurocientíficas sobre las funciones que calculan las neuronas biológicas. Sin embargo, la investigación moderna de redes neuronales se guía por muchas disciplinas matemáticas y de ingeniería, y el objetivo de las redes neuronales no es modelar perfectamente el cerebro. Es mejor pensar en las redes prealimentadas como máquinas de aproximación de funciones que están diseñadas para lograr una generalización estadística, en ocasiones extrayendo algunas ideas de lo que sabemos sobre el cerebro, en lugar de modelos de función cerebral.

Capítulo 3. Análisis de los Datos

En este tercer capítulo realizamos una exposición de los datos, así como un análisis y preprocesado de los mismos, adecuándolos así para la próxima fase.

3.1 Introducción a los datos

A la hora de realizar este Trabajo de Fin de Grado, el primer paso que realizamos fue la búsqueda de un set de datos válido. Durante la búsqueda dimos con dos opciones que se adecuaban a lo que queríamos encontrar.

En primer lugar, trabajando con el ecosistema de Google Cloud Platform, dentro de la herramienta BigQuery, podemos acceder a tablas de prueba que contienen diferentes datos. Una de ellas contiene datos exportados de Google Analytics mostrando analíticas de la página web de Google Merchandise Store[2]. Estos datos contienen información de los visitantes de la página desde el 1 de Agosto de 2016 hasta el 30 de Abril de 2018.

En segundo lugar, navegando por la página web de Kaggle, conocida por sus competiciones de machine learning, encontramos una de ellas relacionada con estos datos y llamada *Google Analytics Customer Revenue Prediction*[3]. Esta competición aportaba los mismos datos, pero en formato CSV. Por desgracia, la competición a la que pertenecen los datos que vamos a utilizar no trata del mismo tema que nosotros. En la competición hemos de predecir el beneficio que aportará cada cliente en los próximos meses a partir de su historial de navegación por el sitio web. Sin embargo, el objetivo de este Trabajo de Fin de Grado es predecir el beneficio por sesión de cliente a partir de los datos de la sesión de navegación actual y su historia previa, prediciendo nosotros la etiqueta, que en este caso será el campo *totals.totalTransactionRevenue*, que muestra el gasto que ha realizado el cliente en dicha sesión.

Dado que los datos son los mismos tanto en Google Cloud Platform como en Kaggle, podemos acceder a ellos de forma diferente según nos ofrezcan más o menos facilidades las herramientas que disponemos.

3.2 Exposición de los datos disponibles

Los datos que poseemos están disponibles en dos formatos: CSV y base de datos de BigQuery. Debido a que el formato original sería el de base de datos de BigQuery, que soporta una estructura de datos de tipo JSON, y que el CSV tiene un formato plano, que deberíamos preprocesar antes de trabajar con ello, hemos decidido utilizar los datos de BigQuery para la exposición de estos

3.2.1 Tablas

En cada conjunto de datos se importa una tabla por cada día de exportación. El nombre de estas tablas tiene el formato “ga_sessions_AAAAMMDD”.

[2] Disponible en https://www.googlemerchandisestore.com/

[3] Disponible en https://www.kaggle.com/c/ga-customer-revenue-prediction/overview

Por otra parte, los datos que se van generando cada día (intradiarios) se importan cada ocho horas. Estos se incluyen en tablas cuyo nombre tiene el formato “ga_sessions_intraday_AAAAMMDD”. A lo largo de un día, cada importación de estos datos sobrescribe la importación anterior en la misma tabla.

Cuando la importación diaria ha finalizado, se suprime la tabla intradiaria del día anterior. Durante las primeras ocho horas de un día, hasta el momento en que se produce la primera importación intradiaria, no hay ninguna tabla disponible. Si se produce un error al crear esta tabla, se conserva la tabla intradiaria del día anterior.

Los datos de un día se consideran definitivos cuando la importación diaria haya finalizado. Es posible que observe diferencias entre los datos intradiarios y los datos diarios en función de las sesiones de usuario activas que sobrepasen el límite de tiempo de la última importación de datos intradiarios.

3.2.2 Filas

Cada fila corresponde a la sesión de un usuario en la tienda *Google Merchandise Store*.

3.2.3 Columnas

A continuación, enumeramos todas las columnas disponibles en el fichero de exportación. En BigQuery, algunas columnas pueden tener campos anidados y mensajes dentro de ellos. Los campos denominados como sección contienen subcampos anidados de información relacionada.

Nombre del campo	**Tipo de datos**	**Descripción**
clientId	CADENA	Versión sin comprimir con tecnología hash del ID de cliente para un usuario determinado asociado a una visita o sesión. Campo obsoleto.
fullVisitorId	CADENA	ID de visitante único (también se denomina ID de cliente).
visitorId	NULL	Este campo está obsoleto. Se usa "fullVisitorId" en su lugar.
userId	CADENA	ID de usuario que se envía a Analytics. Campo obsoleto.

visitNumber	ENTERO	Número de sesión de este usuario. Si es la primera sesión, este valor está configurado como 1.
visitId	ENTERO	Identificador de esta sesión. Es la parte del valor que normalmente se almacena como la cookie _utmb[4]. Solo es único para el usuario. Para obtener un ID completamente único, debe usar una combinación de fullVisitorId y visitId.
visitStartTime	ENTERO	Marca de tiempo (expresada como hora POSIX).
date	CADENA	Fecha de la sesión con el formato AAAAMMDD.
totals	REGISTRO	Esta sección contiene valores globales de la sesión. Por ejemplo: tiempo de visita, número de páginas vistas...
trafficSource	REGISTRO	Esta sección contiene información sobre la fuente de tráfico desde la que se ha generado la sesión. Por ejemplo: tipo de fuente, página web desde la que se ha accedido...
socialEngagementType	CADENA	Tipo de interacción, ya sea "participación desde redes sociales" o "sin participación desde redes sociales".

[4] Cookie ampliamente utilizada para reconocer sesiones de usuarios y diferenciarlas entre ellas, es generada por el código JavaScript propuesto por Google Analytics. Más información: https://developers.google.com/analytics/devguides/collection/analyticsjs/cookie-usage?hl=es-419

channelGrouping	CADENA	Grupo de canales predeterminado asociado con la sesión del usuario final para esta vista.
device	REGISTRO	Esta sección contiene información sobre los dispositivos de usuario. Por ejemplo, sistema operativo; navegador...
customDimensions	REGISTRO	Esta sección contiene las dimensiones personalizadas de usuario o sesión que están configuradas para una sesión. Se trata de un campo repetido y tiene una entrada por cada dimensión que está configurada.
geoNetwork	REGISTRO	Esta sección contiene información sobre la ubicación geográfica del usuario. Por ejemplo, ciudad; país; continente...

Tabla 3.1 Campos de las tablas de exportación de Google Analytics (Google Support, 2018).

3.3 Análisis de los datos

Para el análisis de los datos hemos decidido utilizar en primer lugar BigQuery, debido a su velocidad al realizar peticiones en sets de datos de gran tamaño. Una vez obtenidos los resultados de la consulta, lo hemos convertido a *dataframe*, objeto principal de la librería Pandas, con la que hemos trabajado para su posterior análisis y representación con Plotly.

Para facilitar la lectura de este apartado, primero hemos realizado un análisis por columnas, estudiando de manera independiente cada una de ellas, sin fijarnos en la relación que puede tener con otras. Posteriormente nos adentraremos en las relaciones que pueden guardar y cómo utilizar esas características comunes para tomar diferentes decisiones sobre transformaciones en los datos.

3.3.1 Análisis de columnas

El objetivo de este análisis es razonar qué columnas pueden ser útiles de cara a la elaboración de un modelo predictivo. Hemos realizado un análisis de valores vacíos para ver qué columnas pueden ser descartadas por esta razón. También hemos tenido en cuenta las columnas que según las

especificaciones están obsoletas. Por último, de forma subjetiva, hemos visto el significado de cada columna que queda para decidir cuáles pueden ser de utilidad y cuáles no.

Con el fin de simplificar el análisis, vamos a dividir esta sección según los registros que tenemos en los datos, estudiando en primer lugar las columnas que no pertenecen a ningún registro.

3.3.1.1 Columnas independientes

En primer lugar, hemos realizado un estudio de las columnas sin campos anidados.

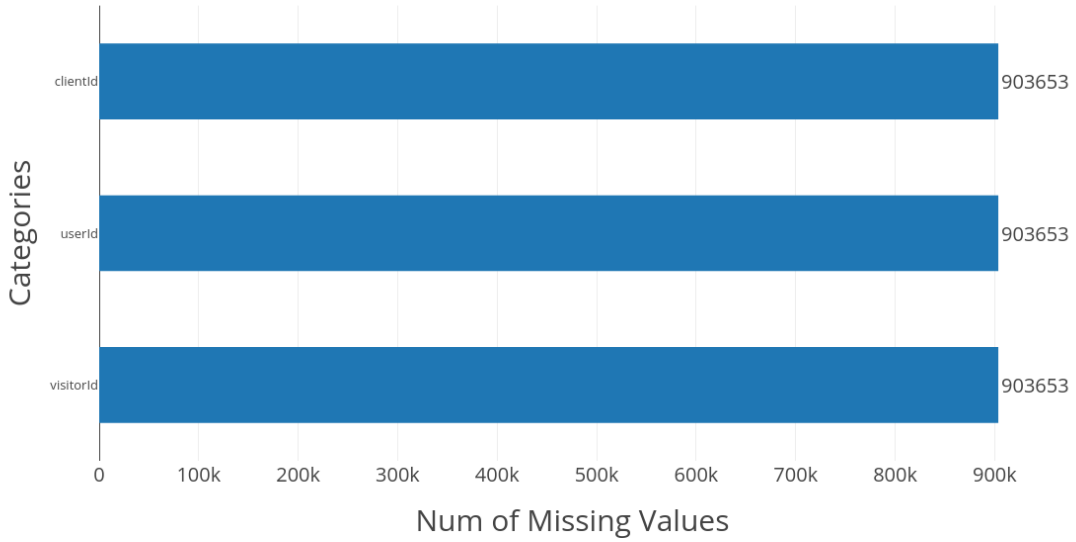


Ilustración 3.1 Valores vacíos en columnas independientes

En el gráfico de la Ilustración 3.1 podemos ver como las tres columnas *clientId, userId* y *visitorId* están completamente vacías (las 903635 filas tienen valor *Nan*), hemos prescindido de ellas.

También hemos prescindido de la columna *socialEngagementType* que, aunque no tiene valores vacíos, tiene siempre un valor constante, algo que no aporta información alguna a los datos.

3.3.1.2 Totals

El primer registro que hemos analizado ha sido el registro *totals*.

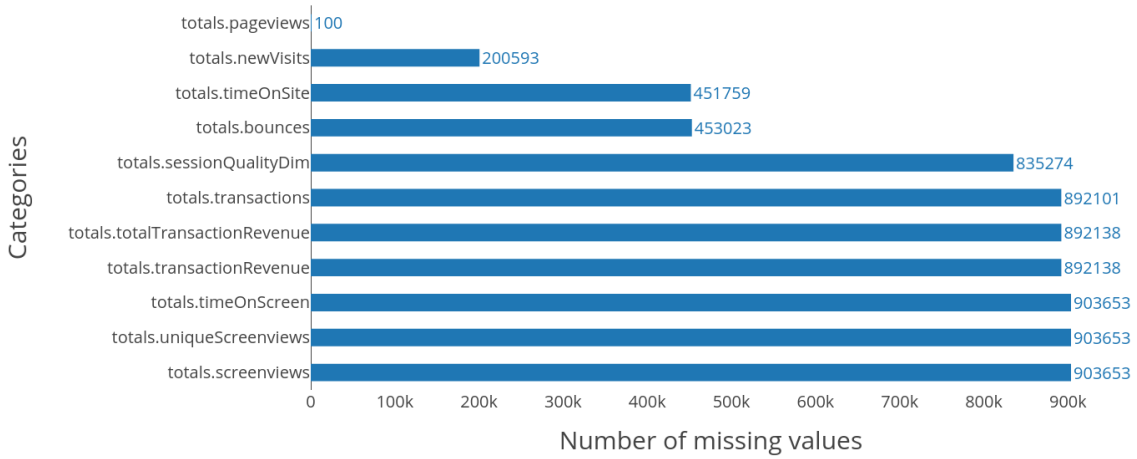


Ilustración 3.2 Valores vacíos en registro totals

Como podemos ver en la Ilustración 3.2, tenemos varias columnas con valores vacíos. Viendo como algunas de ellas tienen demasiados valores vacíos, hemos tomado las siguientes consideraciones:

- *totals.timeOnScreen, totals.uniqueScreenviews* y *totals.screenviews*: Están completamente vacías, lo que implica que las descartamos para nuestro modelo.
- *totals.bounces*: El valor nulo implica que no ha habido rebote, se sustituirá por un 0.
- *totals.timeOnSite*: El valor nulo implica que es 0.
- *totals.newVisits*: El valor nulo implica que no es una nueva visita, se sustituirá por un 0.
- *totals.totalTransactionRevenue*: El valor *null* implica un beneficio de 0€ así que hemos conservado el campo.
- *totals.transactionRevenue*: Es un campo obsoleto según la especificación que nos brinda Google, por ello prescindimos de él.
- *totals.transactions*: El valor *null* implica que no se ha realizado ninguna transacción así que conservamos el campo.
- *totals.pageViews*: El número de valores vacíos es ínfimo, en el análisis por filas veremos si podemos inferirlo o descartaremos esas filas.
- *totals.sessonQualityDim*: Estimador de calidad de la sesión calculado por Google. Desconocemos cómo ha sido calculado, así pues prescindiremos de esta columna.

Por último, al igual que ocurría con la columna *socialEngagementType*, la columna *totals.visits* tiene valor constante, luego la eliminaremos.

3.3.1.3 TrafficSource

Continuando con nuestro trabajo, hemos analizado el registro *trafficSource* que contiene datos de la fuente de la que proviene el tráfico de nuestro sitio web.

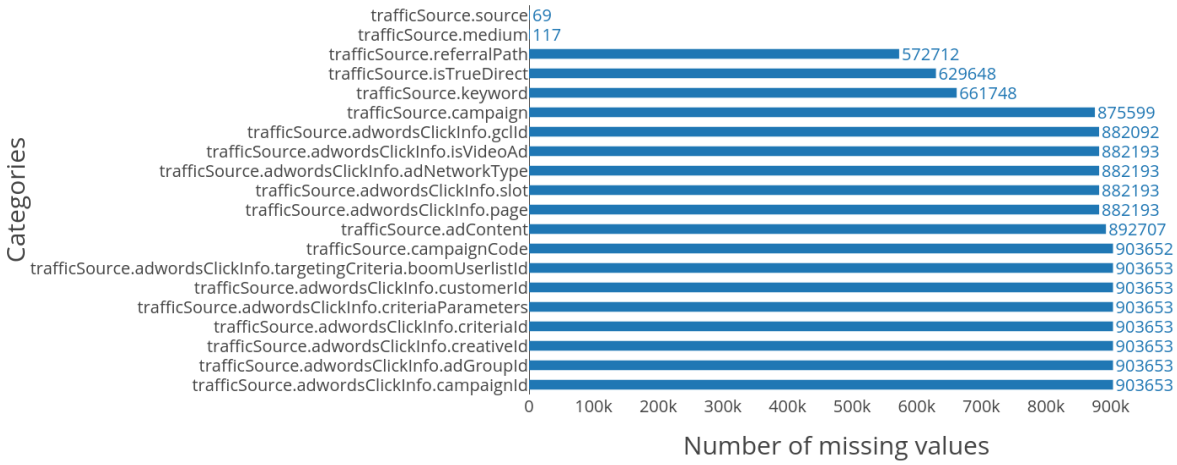


Ilustración 3.3 Valores vacíos en registro trafficSource

En la Ilustración 3.3 podemos ver que algunas columnas tienen demasiados valores vacíos, hemos tomado las siguientes consideraciones:

- *trafficSource.campaignCode*: No contiene información, tienen todos los valores vacíos por lo tanto prescindiremos de las columnas.
- *trafficSource.adContent*: El número de valores vacíos es superior al 98%, descartaremos este campo.
- *traficSource.adwordsClickInfo*.*: El número de valores vacíos es mayor del 97%, descartaremos todos estos campos.
- *trafficSource.keyword*: Solo tiene valor cuando el campo *trafficSource.medium* es "cpc" u "organic search".

  Si nos enfocamos en las filas que cumplen los requisitos que hemos dicho, tenemos un total de 228107 valores vacíos, frente a 19353 con valor. Esto nos deja más de un 92.5% sin valor. Hemos decidido eliminar la columna.

- *trafficSource.isTrueDirect*: Su información es redundante, prescindiremos de esta columna, sólo tiene valor cuando *trafficSource.medium* tiene el valor *"direct"*.
- *trafficSource.referralPath*: Solo tiene valor cuando el campo *trafficSource.medium* es "cpc" u "organic search". Dada su pequeña implicación en los datos, en primera instancia prescindiremos de esta columna.
- *trafficSource.campaign:* Aunque tenga demasiados valores vacíos, sí tiene sentido agrupar todos los datos que no pertenezcan a ninguna campaña, por ello, la falta de valor no implica una falta de información. Hemos utilizado esta columna.

3.3.1.4 Device

Seguimos con el análisis del registro "*device*", el cual contiene información relativa al dispositivo utilizado por los visitantes del sitio web.

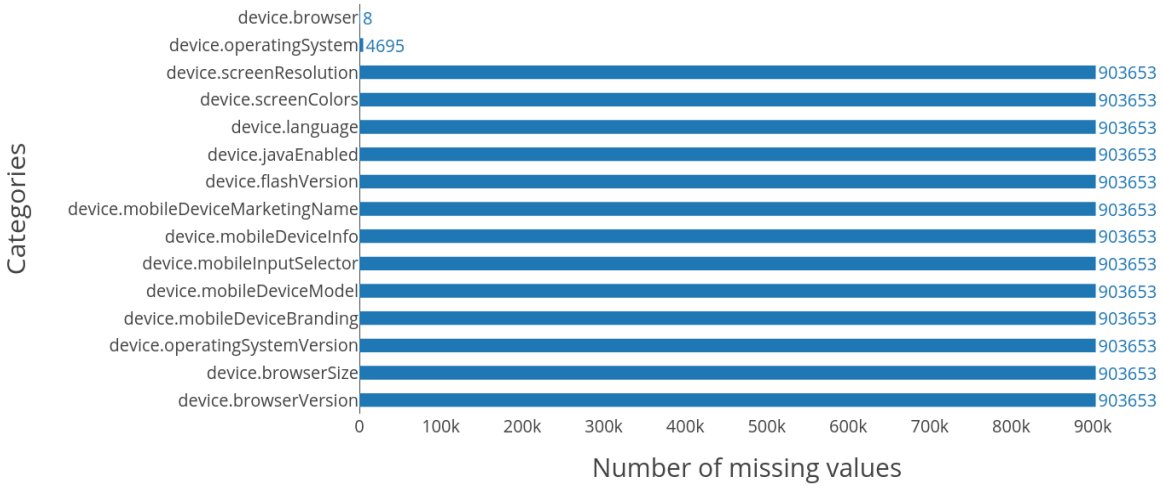


Ilustración 3.4 Valores vacíos en registro device

En la Ilustración 3.4 podemos ver que casi todas columnas tienen demasiados valores vacíos, esto se debe a que Google en su muestra de datos, ha decidido no compartir estas columnas, dado que su valor es "*(not available in demo dataset)*". De esta manera hemos prescindido de todas las columnas, a excepción de *device.browser* y *device.operatingSystem*, que debido al pequeño número de valores vacíos que tienen, en el análisis de filas se decidirá si inferir los datos faltantes o eliminar las filas.

3.3.1.5 GeoNetwork

Por último, procedemos al estudio del registro *geoNetwork*. Este registro contiene campos relativos a la localización del visitante de nuestro sitio web.

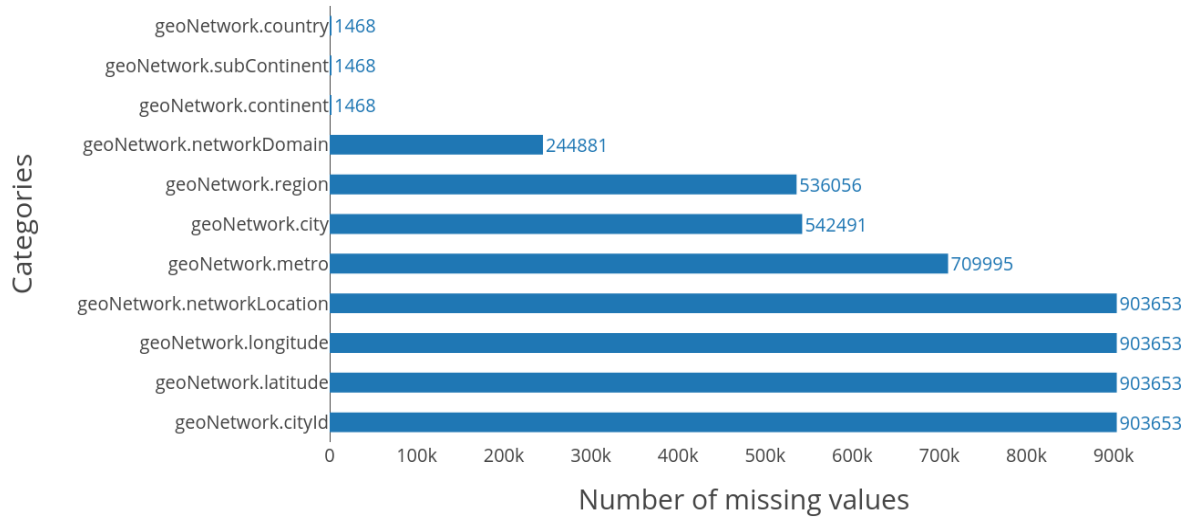


Ilustración 3.5 Valores vacíos en registro geoNetwork

Aunque algunos valores perdidos puedan ser rellenados a partir de los demás, en el caso de las columnas *geoNetwork.cityId*, *geoNetwork.longitude*, *geoNetwork.latitude* y *geoNetwork.networkLocation* no tenemos ningún valor (los 903653 valores están vacíos), por lo tanto serán descartados.

El resto de campos, aunque tienen bastantes valores vacíos, podemos intentar inferir su valor a partir de los demás. Por ejemplo, si tenemos el campo *geoNetwork.city*, pero nos falta el campo *geoNetwork.country*, podemos llegar a inferir su valor.

3.3.2 Análisis por filas

Una vez sabemos que columnas vamos a descartar, podemos considerar que filas pueden sernos de utilidad y cuáles no. En principio, todas las filas son de utilidad, pero si contienen valores vacíos pueden causar problemas en nuestro modelo. Los sets de datos con valores vacíos son incompatibles con algunos estimadores de *scikit-learn*, los cuales asumen que todos los valores del set de datos son numéricos y un significado intrínseco (Scikit-Learn, 2019).

En principio, se pueden manejar dos opciones para afrontar este problema:

- Eliminar las filas con valores vacíos.
- Inferir los valores a partir de diferentes técnicas.

Antes de decidir qué vamos a hacer con cada campo, debemos entender por qué el valor está vacío (Towards Data Science, 2018).

- *Missing at Random* (MAR): Ocurre cuando la propensión a que un valor se pierda no está relacionada con el resto de los datos vacíos, pero sí lo está con algunos de los datos observados.
- *Missing Completely at Random* (MCAR): Es un hecho que el que un valor esté vacío no tiene nada que ver con su valor hipotético ni con el valor de otras variables.
- *Missing not at Random* (MNAR): Las dos posibles razones son que el valor dependa de su valor hipotético (ej. Gente con salarios altos no suele decir su salario en encuestas) o que el valor vacío dependa de otra variable (ej. Los campos *geoNetwork.city* y *geoNetwork.country* están íntimamente relacionados)

En los dos primeros casos, eliminar la fila sería una opción aceptable. Sin embargo, eliminar una fila en el tercer caso puede implicar un desvío en el modelo, afectando a su imparcialidad. También hay que tener en cuenta que inferir los datos no tiene necesariamente que dar un mejor resultado.

De nuevo, hemos realizado el análisis por registros, para facilitar la lectura de esta memoria. En este nuevo paso, recordaremos las columnas que no hemos descartado y que tienen valores vacíos y, según sus características, hemos decidido qué técnica aplicar. Esta vez no hemos hecho mención de los campos independientes dado que todos están completos.

3.3.2.1 Totals

Recordemos que las columnas que hemos decidido conservar son: *totals.bounces, totals.hits, totals.newVisits, totals.pageviews, totals.timeOnSite, totals.totalTransactionRevenue, totals.transactions* y *totals.visits*. En todas ellas el valor *null* implica que tiene valor 0, por lo tanto, sustituiremos el valor directamente sin eliminar ninguna fila.

3.3.2.2 TrafficSource

De este registro hemos mantenido las columnas *trafficSource.campaign, trafficSource.keyword, trafficSource.medium, trafficSource.referralPath* y *trafficSource.source*.

- *trafficSource.campaign*: En este campo el valor *null* implica que no había ninguna campaña asignada, hemos sustituido el valor por una etiqueta "*(not set)*" pues el pertenecer o no a una campaña puede influir en el resultado.
- *trafficSource.medium* y *trafficSource.source*: Estos dos campos los estudiamos en conjunto dada su alto índice de Cramers V[5] de valor 0.867.

  Así pues, hemos decidido rellenar los valores vacíos de *trafficSource.medium* con la moda de cada agrupación según *trafficSource.source*, mientras que si ambos valores están vacíos, hemos eliminado la fila. También hemos comprobado que si *trafficSource.source* está vacío, *trafficSource.medium* también lo está, lo cual nos ha simplificado las cosas.

3.3.2.3 Device

Debemos decidir qué hacer con las filas en las que las columnas *device.browser* y *device.operatingSystem* no tienen valor. Para las filas que no tienen *device.operatingSystem*, debido a la gran cantidad de valores distintos que tiene *device.browser*, no hay una forma fácil de inferir su valor, luego hemos procedido a la eliminación de estas.

No ocurre así con *device.browser*, debido a que todas las filas cuyo valor está vacío, tienen como *device.operatingSystem* el valor "iOS", así pues, hemos rellenado los valores con la moda de este grupo.

3.3.2.4 GeoNetwork

Las columnas importantes de este registro, las cuales hemos conservado, son: *geoNetwork.continent, geoNetwork.subContinent, geoNetwork.country, geoNetwork.region, geoNetwork.metro, geoNetwork.city* y *geoNetwork.networkDomain*.

El estudio de este campo ha sido algo diferente debido a la evidencia de la alta correlación que existe entre los campos, dado que el localizarse en una zona metropolitana implica directamente

[5] El índice de Cramer V representa entre 0 y 1, cómo están relacionadas dos variables categóricas. Más información: https://www.spss-tutorials.com/cramers-v-what-and-why/

encontrarse en una ciudad, país, subcontinente y continente específico.

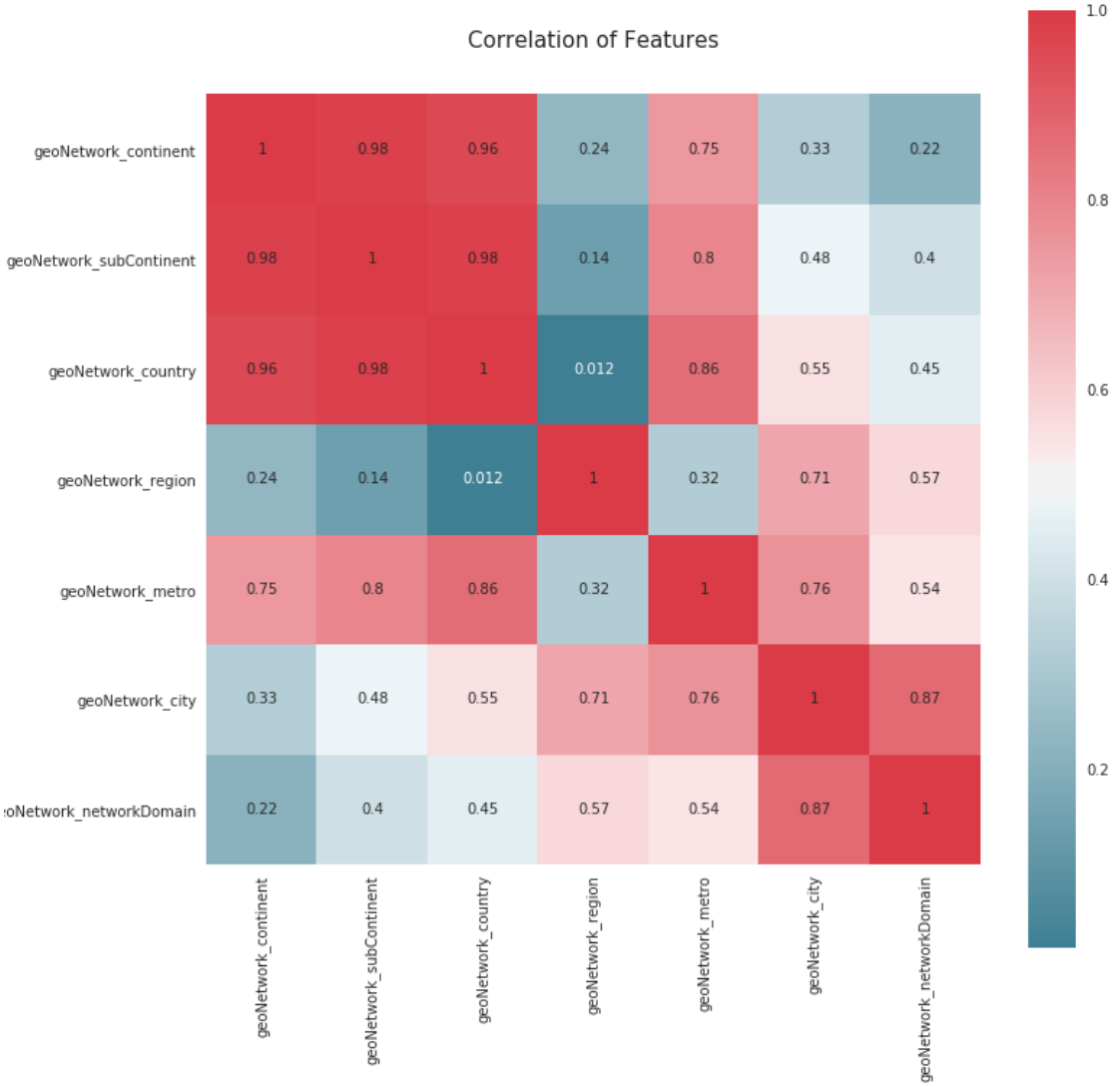


Ilustración 3.6 Matriz de correlaciones del registro geoNetwork

En la Ilustración 3.6 podemos observar la matriz de correlación de estos campos. De esta matriz podemos extraer dos grupos de campos según la correlación que vemos entre estos: el primer grupo, formado por los campos *geoNetwork_city*, *geoNetwork_metro* y *geoNetwork_region*, y un segundo grupo formado por *geoNetwork_country*, *geoNetwork_subContinent* y *geoNetwork_continent*. El campo *geoNetwork.NetworkDomain* lo excluiremos de estos grupos dado que no tiene un significado realmente geográfico, aunque nos será útil en el futuro.

Durante nuestro análisis, hemos observado que es posible que haya incongruencias en los valores de estas columnas. Para comprobarlo, en primer lugar, hemos realizado una agrupación de los datos según el segundo grupo y hemos contado el número de países duplicados[6], obteniendo un resultado de 0 países duplicados. Este resultado implica que no hay conflicto en este primer grupo.

Siguiendo con nuestro análisis hemos comprobado que en este primer grupo, los valores vacíos pertenecen a las mismas filas, imposibilitando la inferencia de unos valores a partir de otros.

[6] Entendemos por país duplicado aquel que pertenezca a diferentes subcontinentes o continentes. Por ejemplo, tener el valor de país “España” en los continentes “Europa” y “África”.

Posteriormente hemos realizado el mismo análisis en nuestro segundo grupo. Hemos visto en primer lugar que para un mismo valor de *geoNetwork.city*, podemos obtener diferentes valores de *geoNetwork.region*, siendo esto una incongruencia geográfica. Entonces hemos decidido sustituir (y rellenar en el caso de valores vacíos) por el valor más repetido del campo *geoNetwork.region* para un mismo valor de *geoNetwork.city*, es decir, con su moda.

A la hora de realizar lo mismo para los campos *geoNetwork.city* y *geoNetwork.metro*, no nos ha sido posible dado que todas las filas con el campo *geoNetwork.city* vacío carecen de valor en el campo *geoNetwork.metro*, imposibilitando la inferencia de este.

Hasta este punto hemos determinado que el atributo *geoNetwork.country* es independiente mientras que el atributo *geoNetwork.region* se puede determinar combinando *geoNetwork.city* y *geoNetwork.region*. Hemos pasado de 6 atributos a 3.

La mayoría de conflictos entre (*geoNetwork.city, geoNetwork.region*) y *geoNetwork.country* se deben a incongruencias en los datos. Esto da lugar a ambigüedades. Dado que no podemos determinar cuál de las filas es la correcta, hemos optado por tomar la combinación más común como la correcta.

Sin embargo, nos parece extraño esta cantidad de errores en el set de datos proporcionado por Google. Hay varias posibilidades, pero la razón que hemos encontrado parece ser que los campos provienen de dos fuentes diferentes, una de la señal GPS del dispositivo y otra la localización del ISP desde el que acceden a internet.

	geoNetwork_city	geoNetwork_region	geoNetwork_country	geoNetwork_networkD...
36216	Madrid	Community of Madrid	Dominican Republic	unknown.unknown
12082	Madrid	Community of Madrid	Portugal	vodafone.pt
21531	Madrid	Community of Madrid	Switzerland	vtx.ch
32308	Madrid	Community of Madrid	Spain	rima-tde.net
32310	Madrid	Community of Madrid	Spain	rima-tde.net

Ilustración 3.7 Filas con datos conflictivos en el registro geoNetwork

En la Ilustración 3.7 podemos ver los diferentes valores que tenemos en las columnas cuando buscamos resultados con el valor “Madrid” en la columna geoNetwork.city. Como podemos observar, el país parece cambiar debido al acceso del ISP. Salta a la vista que, por ejemplo, cuando el país es Portugal, el dominio de la red tiene la terminación *.pt*, lo que nos lleva a pensar que nuestra teoría es correcta.

Aún con todo esto, las columnas *geoNetwork.region*, *geoNetwork.city* y *geoNetwork.metro* tienen más de un 55% de valores vacíos, lo que nos obliga a eliminarlas. Sin embargo, las columnas *geoNetwork.continent*, *geoNetwork.subContinent* y *geoNetwork.country* tienen más de un 99% de valores válidos, luego eliminaremos las filas que no tengan valor en estas 3 columnas.

3.3.3 Creación de características

En este punto, no debemos perder nuestro objetivo, que es predecir el beneficio por **sesión** que vamos a recibir en nuestra tienda online. Nosotros en cada una de nuestras filas tenemos datos de una sesión, independientes unas filas de otras. Sin embargo, no es ninguna locura pensar que el que un cliente acabe comprando en nuestra página no depende solamente de la sesión actual, sino del histórico de sesiones que haya realizado. Por esta misma razón, creemos que es buena idea crear

una serie de nuevas columnas que describan el comportamiento del cliente ligado a la sesión según sus acciones en sesiones anteriores.

Para seguir con el mismo formato de almacenamiento de datos, proveniente de BigQuery, hemos creado un nuevo registro llamado *client* en el que se almacenarán datos totales del cliente de esa sesión:

- *client.transactions*: Número total de transacciones realizadas por el cliente durante todas sus sesiones.
- *client.bounces*: Número total de sesiones rebotadas de este cliente.
- *client.timeOnSite*: Tiempo total que ha pasado el cliente en nuestro sitio web durante todas sus sesiones.
- *client.hits:* Número total de interacciones efectuadas por el cliente.

Debido a que no debemos inferir datos futuros, cuando nos referimos a "durante todas sus sesiones" (del cliente), queremos decir todas sus sesiones pasadas, hasta la actual.

3.3.4 Set de datos final

Finalmente obtenemos un set de datos algo diferente al inicial. En este set de datos final, con el que vamos a trabajar en nuestros modelos, tenemos un total de 23 características y una etiqueta. Las columnas finales son: *channelGrouping, device.browser, device.operatingSystem, fullVisitorId, geoNetwork.continent, geoNetwork.country, geoNetwork.subContinent, totals.bounces, totals.hits, totals.newVisits, totals.pageviews, totals.transactions, totals.timeOnSite, trafficSource.campaign, trafficSource.medium, trafficSource.source, visitId, visitNumber, visitStartTime, totals.totalTransactionRevenue, client.transactions, client.bounces, client.timeOnSite* y *client.hits*, siendo *totals.totalTransactionRevenue* la etiqueta.

Como podemos comprobar, hemos eliminado todas ellas que a nuestro parecer no contenían suficiente información para nuestros modelos, además de crear el nuevo registro *client* e interpretar o inferir los valores vacíos de las ya existentes que hemos conservado.

Capítulo 4. Desarrollo de Modelos

Una vez terminado el análisis y la limpieza de los datos y realizada la introducción teórica vamos a seguir con el desarrollo en código de los modelos mencionados. Para esto vamos a utilizar principalmente la librería SciKit-Learn, en la que podemos acceder a diferentes métodos para realizar nuestros modelos, además de técnicas de normalización y validación.

En este capítulo vamos a utilizar de forma práctica la teoría explicada en el Capítulo 2. Para ello utilizaremos casi exclusivamente clases provenientes de la librería SciKit-Learn, exceptuando la librería Plotly que utilizaremos para la visualización de los resultados.

4.1 Preprocesado

En primer lugar, realizaremos un preprocesado de los datos finales, centrándolos, escalándolos y codificándolos para su correcta interpretación en los estimadores de aprendizaje automático utilizados. Para ello utilizamos las clases *StandardScaler* y *LabelEncoder* de la colección *sklearn.preprocessing*.

- *StandardScaler*: Realiza sobre los datos un escalado estándar, anteriormente explicado.
- *LabelEncoder*: Codifica las clases entre 0 y N-1, donde n es el número de clases. Es muy útil cuando hay un número de clases muy alto y queda descartada la técnica *One-Hot Encoding*[7] debido a la cantidad de columnas que crearía.

Esta decisión puede ser un tanto controvertida, pues el uso de un *LabelEncoder* sólo suele estar recomendado cuando la categoría contiene valores ordinales. Sin embargo, la utilización de *One-Hot Encoder* para según qué columnas es prácticamente imposible debido al gran tiempo de cálculo que nos supondría.

Además, uno de los modelos que vamos a probar se trata de un árbol de decisión, para el cual no suele estar recomendado utilizar un *One-Hot Encoder* dado que induce una gran dispersión de los datos, además de que para un árbol de decisión, las variables son independientes, incluidas las creadas a partir de la codificación (Data Science and Design, 2019).

4.2 Elección de Modelo

Una vez realizado el escalado, centrado y codificación de los datos, podemos proceder a desarrollar los modelos predictivos que creamos convenientes. Para ello, hemos utilizado el *Root Mean Square Error*, la raíz del error cuadrático medio cometido en la predicción del logaritmo natural del beneficio obtenido (en realidad, ln(*totals.totalTransactionRevenue* +1)) . Este método de medida de errores es adecuado para comparar errores de diferentes modelos sobre el mismo set de datos, dado que depende en gran medida de la magnitud de los datos (Hyndman, R. *et al* 2005).

[7] La técnica *One-Hot Enconding* crea tantas columnas como valores únicos tenga la columna fuente, después asigna valor 0 o 1 según si la fila contenía ese valor o no.

En primer lugar, hemos procedido a la comparación de diferentes modelos para ver cuál de ellos puede darnos mejor resultado. En este caso hemos decidido utilizar una validación cruzada de 10 iteraciones, que, como hemos visto en el capítulo 2, divide en 10 subconjuntos el set de datos, entrena con 9 de ellos y evalúa el resultado con el restante. En cada iteración evalúa con un subconjunto diferente, mostrando así 10 resultados de validación.

El problema que tiene la solución anterior es que, tal y como hemos mencionado en el capítulo 2, al estar probando diferentes modelos sobre los mismos datos, estamos presentando un sesgo de selección. Sin embargo, este sesgo de selección no es significativo para conjuntos de datos moderadamente grandes (Kuhn, M. *et al* 2016) ni en un caso como el nuestro en el que hemos comparado un pequeño número de modelos (Abu-Mostafa, Y. *et al* 2012).

Basándonos en los datos de entrenamiento, hemos podido comprobar que, como podemos ver en la Ilustración 4.1, la etiqueta tiene un gran número de valores 0.0. Es decir, hay un gran porcentaje de clientes que no realizan ninguna compra en el sitio web. Por lo tanto, un modelo muy sencillo que podría dar lugar a buenas predicciones sería predecir que el beneficio fuese en todos los casos 0.0. Así pues, también implementaremos este modelo y lo tomaremos como referencia base.

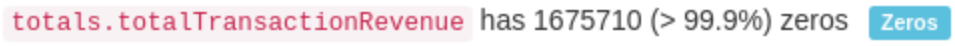


Ilustración 4.1 Número de valores 0.0 que tiene la etiqueta totals.totalTransactionRevenue

Las clases de Scikit-Learn utilizadas son: *SGDRegressor* con sus parámetros por defecto exceptuando el parámetro *penalty*, el cual tunearemos para realizar una regresión lineal con valor *none*, una regresión Lasso con valor *l1*, una regresión Ridge con valor *l2* y una regresión ElasticNet con valor *elasticnet*, todas ellas con la técnica de descenso estocástico de gradiente; SVR, con sus parámetros por defecto, para realizar una regresión de máquina de soporte vectorial con kernels de función de base radial; un árbol de decisión para regresión con la clase RegressionTree con sus parámetros por defecto; *MLPRegressor* para diferentes redes neuronales prealimentadas, cambiando el número de capas ocultas y neuronas por capa[8] gracias al parámetro *hidden_layer_sizes* y por último *DummyRegressor* para hacer el modelo que predice siempre 0.0 para cualquier entrada.

Estos modelos han sido comparados con el método *cross_val_score* que nos provee la librería. Este método ejecuta de forma automática la técnica de validación cruzada y nos devuelve un array con los resultados de validación que hemos conseguido en cada iteración.

Antes de realizar el modelado, debemos acondicionar los datos. En primer lugar se ha utilizado la clase *LabelEncoder* para codificar las variables categóricas, dado que la entrada de los modelos han de ser numéricas. Este método hace una asignación uno a uno sobre los diferentes valores de la columna en cuestión y una serie ordinal de números (1, 2, 3, etc…).

Después de la codificación de las columnas categóricas, debemos utilizar la clase *StandardScaler* para el centrado y escalado de los datos numéricos. Esto se realiza para: reducir la distancia entre valores gracias al escalado y poder explotar la simetría de los datos (aumenta la velocidad de convergencia de algunos modelos) debido al centrado en 0.

[8] Seguiremos la notación NeuralNetwork_tamaño1_tamaño2_..._tamañoN, siendo N el número de capas y tamañok el número de neuronas de la capa k-ésima.

Ahora bien, no podemos realizar el escalado sobre todos los datos si después vamos a realizar la validación cruzada, dado que estaríamos utilizando datos de validación para transformar los de entrenamiento. SciKit-Learn nos provee de una clase llamada *Pipeline* que nos permite juntar la transformación de datos y el modelo en una sola instancia que introducimos como parámetro en la función *cross_val_score*. Sin embargo, esto se realiza sobre todas las columnas, cuando nosotros solo queremos que se realice sobre las columnas numéricas. Para ello hemos hecho uso de la clase *ColumnTransformer* en la que podemos introducir uno o más transformadores, en orden de ejecución, y las columnas a las que afectará cada uno. En nuestro caso el *ColumnTransformer* solamente tendrá un paso, el *StandardScaler*.

Teniendo en cuenta lo anteriormente descrito, hemos realizado la comparación de los diferentes modelos y hemos obtenido los siguientes resultados:

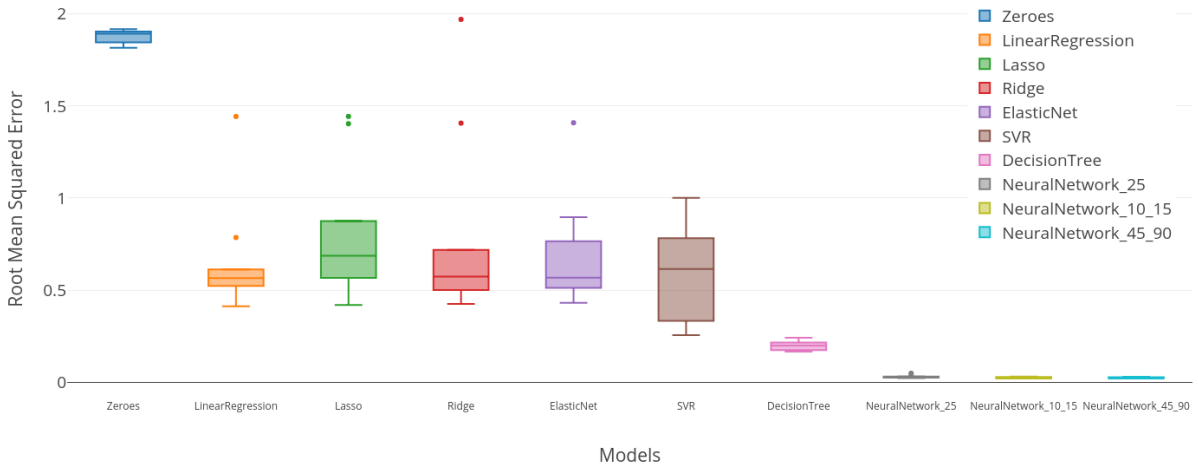


Ilustración 4.2 Comparativa de modelos en gráfico de cajas con métrica de error Root Mean Squared Error

Como vemos en la Ilustración 4.2, todas las predicciones mejoran el resultado del *DummyRegressor*, el nombrado como *Zeroes*, el cual tomábamos como referencia base. Sin embargo, podemos observar que las redes neuronales tienen un mejor desempeño. Éste es el mejor modelo por varios factores, entre ellos, podemos observar como la mediana es la menor de todas y no posee outliers; la varianza es mínima luego sus cuartiles tienen un tamaño muy pequeño y por último, posee los valores más bajos para la medida de error. Al juntar todas estas características, podemos ver claramente como es la mejor elección para usar como modelo.

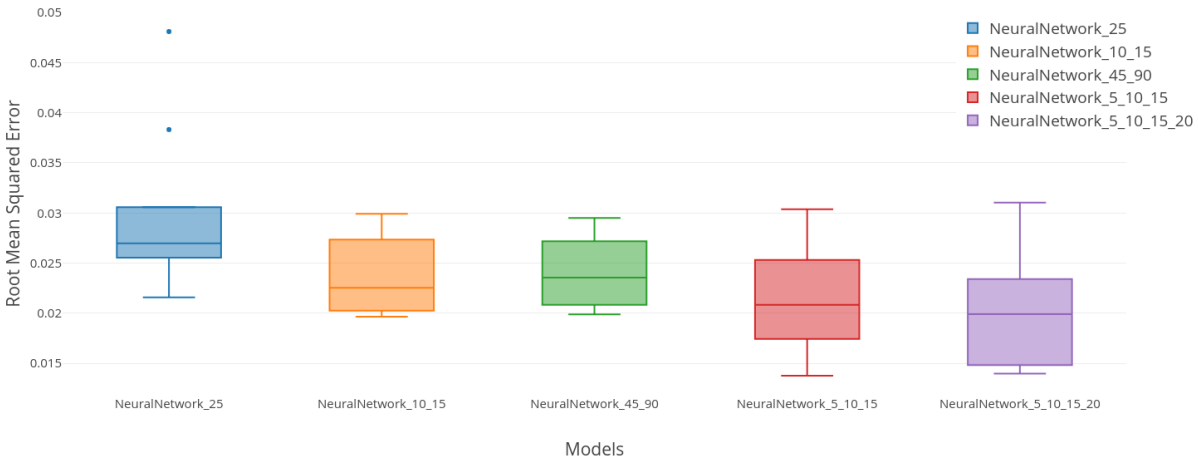


Ilustración 4.3 Comparativa de Redes Neuronales en gráfico de cajas con métrica de error Root Mean Squared Error

En la Ilustración 4.3 Comparativa de Redes Neuronales en gráfico de cajas con métrica de error Root Mean Squared ErrorIlustración 4.3 podemos ver más nítidamente la comparativa de las redes neuronales prealimentadas. Hemos desarrollado 4 de ellas, tuneando el modelo cambiando el número de capas ocultas y el tamaño de las mismas. Hemos obtenido un buen resultado aplicando todas ellas, pero podemos ver como aumentando el número de capas, aumenta la precisión en la mayoría de los casos. De esta manera, la red neuronal que mejor resultado ha dado es la red de 4 capas con 5, 10, 15 y 20 neuronas respectivamente.

Capítulo 5. Conclusiones y Líneas Futuras

El uso de este tipo de técnicas de aprendizaje automático supone un cambio de paradigma. Nosotros partimos de unos datos y un resultado para hallar la función que los relaciona, en lugar de tener unos datos y una función y hallar el resultado. Este tipo de técnicas se han vuelto muy atractivas en los últimos años debido a sus excelentes resultados en diferentes campos.

En la actualidad, la aplicación de este tipo de técnicas implica una mejora significativa en multitud de campos, desde negocios, sanidad, econometría… Nosotros solo hemos hecho uso de algunas técnicas sencillas en la realización de este Trabajo de Fin de Grado. Además, al ser un campo en desarrollo, día a día se crean nuevos avances que pueden mejorar los resultados.

5.1 Conclusiones

El principal objetivo de este Trabajo de Fin de Grado es predecir el beneficio por cliente de la Google Merchandise Store, basándonos en las sesiones de clientes realizadas anteriormente aplicando sobre ellas algoritmos de aprendizaje automático para obtener una función con la que, introduciendo como entrada la sesión completa de un cliente actual, obtengamos un resultado que se corresponde con el beneficio que vamos a obtener por ese cliente en la sesión dada.

Durante el desarrollo de este Trabajo de Fin de Grado hemos realizado diferentes transformaciones de los datos que han resultado cerca del 80% del trabajo realizado, y después hemos elegido, creado y entrenado un modelo que nos ha llevado el 20% del tiempo. Esta distribución de tiempo es habitual en la ciencia de datos (KDNuggets, 2019).

El mejor resultado que hemos obtenido ha sido con una red neuronal y ha sido de 0.014901 de *Root Mean Squared Error* siendo un valor bastante bajo. El último paso a dar, sería entrenar el modelo con todos los datos que disponemos y utilizarlo en casos reales, en los que debería tener un buen desempeño.

Dejando de lado los resultados, durante el desarrollo de este Trabajo de Fin de Grado, hemos tenido la oportunidad de aprender estas novedosas técnicas, que se utilizan ampliamente en la actualidad. El aprendizaje ha partido prácticamente desde cero, ya que la base de la que partíamos era un conocimiento de la parte más básica de Python. Sobre esta base hemos aprendido los fundamentos de aprendizaje automático, a analizar datos con Pandas, visualizarlos con Plotly y Matplotlib, ejecutando todo esto sobre una instancia en la nube de Google Cloud Platform. También hemos aprendido a extraer datos de bases de datos de BigQuery (aprovechando las facilidades que nos ofrece GCP), y a desarrollar modelos predictivos con la librería SciKit Learn.

Todo este conocimiento lo hemos ido adquiriendo paso a paso según los requerimientos del objetivo principal del Trabajo de Fin de Grado. Es cierto que en la última parte de este documento, tratada en el capítulo 4, hemos comparado modelos muy sencillos con los parámetros por defecto. Esto, sin embargo, se ha debido al limitado tiempo del que hemos dispuesto y lo comentaremos en las líneas futuras.

5.2 Líneas futuras

Una vez cumplido el principal objetivo de este Trabajo de Fin de Grado, podemos plantear diferentes opciones descartadas, ya sea por el tiempo del que hemos dispuesto o por el limitado poder de computación de nuestra máquina.

La primera posibilidad hubiese sido crear más características, fijándonos en el tiempo. Al crear el registro cliente hemos tenido en cuenta todas las interacciones pasadas del cliente en sus sesiones, dando igual importancia a una sesión de hace un año como a una de hace 2 días. Seguramente crear diferentes registros cada cierto tiempo podría mejorar el resultado, por ejemplo, teniendo en cuenta las sesiones de la última semana, los últimos 3 meses, el último año y desde el principio de los tiempos.

Otra posibilidad, debido a que la gran mayoría de los casos tienen un beneficio nulo, es decir, el valor de la etiqueta es 0, podríamos aplicar un modelo en dos fases. Este modelo constaría de una primera clasificación de compra/no compra seguida de una regresión para los datos que pasen el primer filtro, dejando todos los que se predigan con etiqueta no compra con valor 0. Además, como la variación numérica es grande, aplicaríamos un logaritmo (como ya hacemos) para normalizar los datos.

Por último podríamos considerar el utilizar modelos más complejos. En este Trabajo de Fin de Grado hemos indagado en modelos simples como son una regresión lineal, un árbol de decisión o una máquina de soporte vectorial. Sin embargo, podríamos recurrir a modelos más complejos, incluso en algunos basados en estos anteriores, como pueden ser un Random Forest, redes neuronales más complejas o LightGBM, la usada en el kernel ganador de la competición de Kaggle (Kaggle, 2019), además de elegir los parámetros más adecuados para cada modelo (apoyándonos de nuevo en la validación cruzada) en lugar de utilizar simplemente los valores por defecto.

REFERENCIAS

Amplitude: *The Early Days of Web Analytics*. Disponible en: https://amplitude.com/blog/2015/06/15/the-early-days-of-web-analytics Junio 2019

Dealipedia: *SPSS acquires NetGenesis for $44.6 million*. Disponible en: http://www.dealipedia.com/deal_view_acquisition.php?r=2513 Junio 2019

I. Gorostiza, A. Barainca, *Google Analytics, Mide y Vencerás.* España: Editorial Anaya, 2016

M. Miller, *Sams Teach Yourself Google Analytics in 10 Minutes.* United States: Sam Publishing, 2011

Data Science Project Management: *CRISP-DM and KDD*. Disponible en: http://www.datascience-pm.com/crisp-dm-and-kdd/ Junio 2019

Kaggle: *Google Analytics Customer Revenue Prediction: Predict how much GStore customers will spend*. Disponible en: https://www.kaggle.com/c/ga-customer-revenue-prediction Junio 2019

Google: *Esquema de BigQuery Export*. Disponible en: https://support.google.com/analytics/answer/3437719?hl=es Junio 2019

Google: *Bigquery Cookbook*. Disponible en: https://support.google.com/analytics/answer/4419694 Junio 2019

S. Cooper, *Data Science from Scratch*. United States: Data Science, 2018

Towards Data Science: *How to Handle Missing Data*. Disponible en: https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4 Junio 2019

Medium: *Filtering csv files bigger than memory to a pandas dataframe*. Disponible en: https://medium.com/@vincentteyssier/filtering-csv-files-bigger-than-memory-to-a-pandas-dataframe-3ab51ff993fd Junio 2019

J. Hurwitz, D. Kirsch, *Machine Learning for Dummies,* United States: IBM, 2018

I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning Book*. United States: MIT Press, 2016

S. Russel, P. Norvig, *Artificial Intelligence: A Modern Approach*. United States: Pearson, 2016

S. Skiena, *The Data Science Design Manual*, United States: Springer, 2017

M. Kuhn, K. Johnson, *Applied Predictive Modeling*, United States: Springer 2016

Y. Abu-Mostafa, M Magdon Ismail, H. Lin, *Learning from Data: A Short Course*, United States: AMLBook, 2012

A. Burkov, *The Hundred-Page Machine Learning Book*, United States: Disponible en: http://themlbook.com/ Julio 2019

R. Berk, *Statistical Learning from a Regression Perspective*, United States: Springer, 2008

G. Bonaccorso, *Mastering Machine Learning Algorithms*, United Kingdom: Packt Publishing, 2018

Medium: Machine Learning: *Como desarrollar un modelo desde 0.* Disponible en: https://medium.com/datos-y-ciencia/machine-learning-c%C3%B3mo-desarrollar-un-modelo-desde-cero-cc17654f0d48 Julio 2019

Towards Data Science: *Scale, Standarize or Normalize with SciKit Learn*. Disponible en: https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02 Julio 2019

Towards Data Science: *A Beginners Guide to Linear Regresion in Python with SciKit Learn.* Disponible en: https://towardsdatascience.com/a-beginners-guide-to-linear-regression-in-python-with-scikit-learn-83a8f7ae2b4f Julio 2019

Iartificial: *Regularización Lasso L1, Ridge L2 y Elasticnet.* Disponible en: https://iartificial.net/regularizacion-lasso-l1-ridge-l2-y-elasticnet/#Regularizacion_Lasso_L1 Julio 2019

E. Carmona, *Tutorial sobre Maquinas de Vectores de Soporte (SVM)*, Disponible en: http://www.ia.uned.es/~ejcarmona/publicaciones/[2013-Carmona]%20SVM.pdf Julio 2019

S. Gunn, *Support Vector Machines for Classification and Regression*, United Kingdom: University of Southampton, 1998

A. Smola, *Regression Estimation with Support Vector Learning Machines*, Disponible en: https://www.researchgate.net/publication/2879212_Regression_Estimation_with_Support_Vector_Learning_Machines Julio 2019

A. Smola, *A Tutorial on Support Vector Regression*, Disponible en: https://alex.smola.org/papers/2004/SmoSch04.pdf Julio 2019

Numerentur: Máquina de Soporte Vectorial. Disponible en: http://numerentur.org/svm/ Julio 2019

L. Breiman, J. Friedman, R. Olshen, C. Stone, *Classification and Regression Trees*, United States: Chapman and Hall, 1998

S. Sayad, *Decission Tree – Regression*, Disponible en: https://www.saedsayad.com/decision_tree_reg.htm Julio 2019

Machine Learning Mastery: *Classification and Regression Trees for Machine Learning*, Disponible en: https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/ Julio 2019

Medium: *Visiting Categorical Features and Encoding in Decision Trees.* Disponible en: https://medium.com/data-design/visiting-categorical-features-and-encoding-in-decision-trees-53400fa65931 Julio 2019

R. Hyndman, A. Koehler, *Another look at measures of forecast accuracy*. Disponible en: https://robjhyndman.com/papers/mase.pdf

Towards Data Science: *Boosting Algorithm GBM*. Disponible en: https://towardsdatascience.com/boosting-algorithm-gbm-97737c63daa3 Agosto 2019

Sitio Big Data: *Machine Learning: Métricas de Regresión MSE*. Disponible en: http://sitiobigdata.com/2018/08/27/machine-learning-metricas-regresion-mse/# Agosto 2019

Kaggle: *Google Analytics Customer Revenue Prediction Winning Solution.* Disponible en: https://www.kaggle.com/kostoglot/winning-solution#L144 Agosto 2019

KDNuggets: *Pareto Principle for Data Sciencist.* Disponible en: https://www.kdnuggets.com/2019/03/pareto-principle-data-scientists.html Agosto 2019

ANEXOS

ANEXO I

Preparación previa

En este anexo se incluyen todos los pasos que hemos seguido antes de comenzar la realización de este Trabajo de Fin de Grado. Estos pasos se han realizado para la correcta preparación y adquisición de las habilidades necesarias para afrontar el problema planteado.

5.3 Cursos

Para la realización de este Trabajo de Fin de Grado, hemos realizado cursos de dos fuentes:

- Google para la comprensión de los datos y la herramienta Google Analytics.
- Kaggle para la programación en python con librerías relacionadas con Machine Learning.

5.3.1 Curso de introducción a Google Analytics

Este curso se realizó para conocer la herramienta de la que proceden los datos que hemos empleado. En el curso, que consta de 17 lecciones agrupadas en 4 unidades, podemos adquirir un conocimiento global del uso de la herramienta:

- Cómo funciona Google Analytics y cómo configurarlo para un sitio web.
- Cómo navegar por su interfaz, descargar informes y configurar el panel de control.
- Comprensión de los tres reportes fundamentales: Informes de audiencia, informes de adquisición e informes de comportamiento.
- Conceptos básicos sobre las campañas y conversiones.

5.3.2 Curso avanzado de Google Analytics

Este segundo curso se realizó para profundizar en la herramienta, dado que el curso anterior nos pareció insuficiente. En la realización de este curso, no solo hemos entrado más en detalle sino que hemos afianzado conocimientos del curso anterior. El curso consta de 18 lecciones agrupadas en 4 unidades en las que podemos ver:

- Procesamiento, recogida y clasificación de los datos.
- Ajuste de configuración con métricas y dimensiones personalizadas
- Técnicas de análisis avanzado y segmentación de datos.
- Herramientas de márketing avanzadas.

5.3.3 Curso de programación en Python

Este curso de 7 lecciones sobre el lenguaje de programación Python, nos ha ayudado a conocer uno de los lenguajes más utilizados en el ámbito del aprendizaje automático. En este curso hemos aprendido cosas básicas del lenguaje.

5.3.4 Curso introducción al Machine Learning

En este curso de 7 lecciones, hemos aprendido, de forma teórica, diferentes cosas:

- Cómo funciona un modelo de Machine Learning
- Cómo realizar una exploración básica de los datos
- Técnicas de validación y normalización para combatir el overfitting y el underfitting.

5.3.5 Curso intermedio de Machine Learning

Este curso de 7 lecciones es complementario al anterior y nos permite ahondar en técnicas más complejas. Como el curso anterior, este curso es teórico y hemos aprendido cosas como:

- Tratamiento de valores vacíos
- Tratamiento de variables categóricas
- Tuberías, validación cruzada y xgboost.
- Maneras de evitar fuga de datos que pueden estropear nuestro modelo.

5.3.6 Curso de Pandas

En este curso de 6 lecciones hemos aprendido a utilizar la librería de python más importante en lo que al tratamiento de datos se refiere. En este curso hemos aprendido a manejar dataframes y series, las unidades de datos de pandas, realizando desde agrupaciones y ordenaciones hasta lecturas, escrituras y transformaciones de los datos.